EVALUATING HEAD GESTURES FOR PANNING 2-D SPATIAL INFORMATION

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Matthew Oliver Derry

December 2009

COMMITTEE MEMBERSHIP

TITLE:	Evaluating Head Gestures For Panning 2-D Spatial Information
AUTHOR:	Matthew Oliver Derry
DATE SUBMITTED:	December 2009
COMMITTEE CHAIR:	Dr. Franz Kurfess
COMMITTEE MEMBER:	Dr. Gene Fisher
COMMITTEE MEMBER:	Dr. Clark Turner

ABSTRACT

Evaluating Head Gestures for Panning 2-D Spatial Information

by

Matthew Oliver Derry

New, often free, spatial information applications such as mapping tools, topological imaging, and geographic information systems are becoming increasingly available to the average computer user. These systems, which were once available only to government, scholastic, and corporate institutions with highly skilled operators, are driving a need for new and innovative ways for the average user to navigate and control spatial information intuitively, accurately, and efficiently. Gestures provide a method of control that is well suited to navigating the large datasets often associated with spatial information applications. Several different types of gestures and different applications that navigate spatial data are examined. This leads to the introduction of a system that uses a visual head tracking scheme for controlling of the most common navigation action in the most common type of spatial information application, panning a 2-D map. The proposed head tracking scheme uses head pointing to control the direction of panning. The head tracking control is evaluated against the traditional control methods of the mouse and touchpad, showing a significant performance increase over the touchpad and comparable performance to the mouse, despite limited practice with head tracking.

ACKNOWLEDGEMENTS

I would like to thank my thesis advisor, Dr. Kurfess for his willingness to take me on as his advisee when I really needed it. His feedback throughout this process was always insightful and made this thesis what it is today. I also want to extend a hearty thank you to Dr. Gene Fisher and Dr. Clark Turner for joining my thesis committee on short notice without even so much as a grumble about the inconvenience.

I would also like to thank Karen Redwine for her advising on the statistical analysis of the results of this research. Without her help I would still be trying to figure out what an ANOVA is. I would also like to thank Jan Jaroncyk for her extensive effort in helping me to reign in my tendency towards verbosity. Finally, I would like to thank all my friends and family who supported me or contributed to this research in some way. Thank you!

TABLE OF CONTENTS

List of Tables

List of Figures

1. Introduction

Due to the large quantity of data frequently associated with spatial information, systems dealing with such information inherently have challenging technical requirements that must be tackled for the system to be useful. With ever increasing computing power, bandwidth, and more sophisticated data collection, storage, and retrieval techniques, many of those challenging technical requirements are being addressed. Consequently, more spatial information systems are becoming available to the general public. New, often free, applications such as mapping tools, topological imaging, GIS (Geographic Information Systems), which were once available only to government, scholastic, and corporate institutions with highly skilled and trained operators, are now driving a need for new and innovative ways to navigate and control spatial information intuitively, accurately, and efficiently so that the average computer user may make full use of these tools.

The most common methods for control of these tools revolve around a mouse and keyboard, or in the case of laptops, a touchpad and keyboard. Because the space for actually moving a mouse or engaging a touchpad is limited, continuous panning or scrolling requires constant resetting of the placement of these input devices. Unfortunately, due to the size of the datasets of spatial data (i.e. large maps of cities, states, or countries), continuous panning is a very common requirement for navigating these types of systems [21]. While it is possible to create a scheme for continuous panning using a mouse or touchpad, it is not something that is commonplace in current applications. With these drawbacks, the natural question arises, is it possible to augment

the keyboard and mouse approach by adding another mode of input to handle panning control which is as intuitive, efficient, and accurate as a mouse or touchpad?

Gestures can provide an intuitive mode of input that allows for controlling the panning component of navigating spatial data, which could be picked up relatively easily. Additionally, head gestures, or more specifically, head turn has a natural correlation with the notion of panning in spatial navigation. That is to say, panning in the direction of gaze is a natural and intuitive action for controlling a spatial information system [25]. Another benefit of head gestures is that fatigue would not be as much of an issue as it would with other types of gestures. Head gestures can, in many cases, be useful for individuals with certain disabilities in which their hands cannot be used for control. Additionally, there are other domains where hands-free navigation of spatial data could be beneficial. Consider surgeons using head turn gestures to move an arthroscopic camera, freeing their hands to control their arthroscopic tools.

The goal of this research is to examine the practical considerations, as well as the usefulness of just such an augmentation in navigating spatial data. Specifically, a system utilizing a common webcam tracks the location and rotation of a user's head to control the continuous panning in the 2D mapping application, Google Maps. The general suitability of head gestures for navigation tasks, and the performance of users for a few scenarios are examined in the experimental part of this thesis. The users are given a series of panning-specific tasks using a mouse, a touchpad, and head gestures to control the panning. The tasks are timed and the methods are comparatively ranked for accuracy.

The results of the experiments are evaluated to identify advantages and drawbacks of using head gestures for navigation compared to conventional navigation methods.

2. Previous Work

2.1 Gestures used in Human-Computer Interaction

Gestures have been used by humans for thousands of years, to both augment verbal communication, *e.g.* pointing to something while asking for it, or replace verbal communication all together with a sign language, such as American Sign Language. Both the expressive power and the intuitive and universal nature of gestures has lead to a significant amount of research on using gestures to communicate with, and control, computer applications.

There are several options available to an interface designer when choosing which gestures to include in creating controls for computer applications. Those gestures include hand gestures, arm gestures, full-body gestures, facial expressions, and head gestures [21, 31, 41, 48, 49]. What follows is a discussion of these different gesture types along with their benefits and drawbacks.

Hand gestures are any kind of movement or pose done with just the hand. This has several advantages in that the hand is a relatively simple object to recognize using computer vision techniques. Additionally, because the hand has a high level of dexterity, a wide range of shape and motion combinations are possible, this is illustrated by the fact that there is an entire sign language based on the motions and positions of hands. This lends itself well to controlling a tool with many different commands [29]. One downside to using hand gestures is the possibility for fatigue and possible injury, *e.g.* Carpal Tunnel Syndrome, with extended usage and repetitive motion. Another downside to using hand gestures for control is that if the hand is occupied with the task of controlling the

program, the user is limited in using it for other aspects of control [48]. This isn't so much a problem if the hand is responsible for controlling a lot of things, but if it is responsible for just a few actions, then the dexterity of the hand is wasted. Finally, a fundamental weakness of hand gestures is that the control vocabulary must be easy to recreate as well as remember, limiting the overall control vocabulary available to the system designer [13]. Overall, the hand is a good tool for control, illustrated by the fact that the standard mouse and keyboard controls function with what are essentially hand gestures.

Arm gestures fall somewhere between hand gestures and full-body gestures. They are especially suited for tasks involving pointing [46] or tasks in which the lower body is not involved, such as in the suite of sports games for the Nintendo Wii. They can be very expressive but are often not as nuanced as hand gestures. Fatigue can be a real issue with arm gestures, especially if the control gestures require the user to hold their arms away from their body either for long periods or with high frequency. Several systems use arm gestures for control [4, 46, 52], and they can be quite intuitive, but they are better suited for environments with large displays, where the full range of arm motion can be taken advantage of. As an aside, hand gestures and arm gestures are often combined for control.

Full-body gestures are another type of gesture used for human-computer interaction [48]. With this approach, the whole body is used to complete poses and gestures as the control. Like hand gestures, full-body gestures offer many different poses and gestures for control. The difficulties with full-body gestures are the same as the

difficulties with hand gestures, only they are magnified. Fatigue becomes more of an issue because the whole body is involved in the process [6]. Additionally, all parts of the body are used for the one control, thereby limiting the user to just one mode of input. Full-body gestures are not very compatible with a system in which the user is sitting in front of a terminal. They can, however, be useful in a virtual world or virtual reality setting where the user is moving around in a simulated environment.

Finally, there are facial expressions and head gestures. While heads are relatively easy to pick out of an image, facial expressions are much harder to discern using current approaches [23]. Additionally, as a control, facial expressions are limited due to small number of discernible expressions, as well as their sometimes subconscious nature. Due to these factors, facial expressions are not considered in this research. On the other hand, head movements and gestures are relatively easy to pick out of an image using established techniques [3, 27, 30, 36]. Certain head motions can be done for long stretches without fatigue or strain. Additionally, several head gestures are very intuitive, *e.g.* look left to pan left. look right to pan right, etc. Finally, head gestures correspond well with continuous action. This eliminates the repeated reseting that occurs with navigation methods such as the touchpad or mouse when the available physical work space isn't large enough to accommodate the on-screen task. *E.g.* panning a map that is bigger than the screen and the touchpad space, so the panning can only go as far as the size of the touchpad in a single swipe. Despite these advantages, head gestures suffer from some drawbacks as well. There are very few gestures available using head gestures. Out of what few gestures there are, there are even fewer that remain comfortable over

extended periods of use. For instance, tilting the head towards the shoulder quickly becomes uncomfortable when done for any significant amount of time; this is because the head is moved out of alignment with the spine, causing the neck muscles to bear much of the weight [26]. Another issue that should be considered when using head gestures is that large screens will require users to turn their heads farther than can be compensated for by their eye movements, thereby effectively creating blind spots on certain parts of the display [7]. Despite these drawbacks, the use of head rotation for panning fits this research very well. Head turn is one of the few motions that can be done for long periods of time by able-bodied people, and on the majority of displays in use today by a typical user, the large display issue will not be a problem. For these reasons, this research focuses on head turn to control the panning component of the navigation of spatial data.

All of the systems using gestures for control suffer from the problem of having to identify intentional gestures versus unintentional gestures. While much work has been done in an attempt to automate this recognition, [1, 20, 39, 42] it is a difficult problem and to this point lacks a definitive solution. Currently, this issue is primarily addressed by providing the user with some method to indicate to the system whether or not the gesture is intentional or unintentional.

2.2 Input Methods for Capturing Gestures

There are two primary methods for capturing gestures as input for a computer program. The first such method is a sensor-based approach [49]. A sensor-based approach is one where the gestures are captured using some sort of sensor placed on the

user's body. These sensors can be anything from infrared transmitters placed on gloves to accelerometers placed on glasses or a visor that tracks the movement of a user's head. Several systems, in wide-spread use today, employ this sensor-based approach. One such system is the Apple iPhone®, where an accelerometer is embedded in the iPhone, which is in the user's hand. With the accelerometer it has the ability to switch between landscape viewing mode and portrait viewing mode simply by turning the device on its side. Another such system in widespread use is the Nintendo Wii®, which uses accelerometers in its controller to capture gestures for controlling different games such as the bowling or baseball games. Mattell was one of the first video game companies to attempt to bring gesture-based control to the mass market with an early attempt at using gestures to control games on the original 8-bit Nintendo system called the Power Glove [49]. This glove used ultrasonic transmitters that sent signals to receivers that were placed on the TV. A benefit to utilizing sensors actually attached to the user is that they are generally very precise. Consequently though, special hardware is often required for the control to be used making it more difficult to distribute the tool for widespread use.

The other method for capturing gestures is a visual approach. For this approach the gestures are interpreted by the system from a video stream of one or more cameras [5, 19, 45]. There are many systems where multiple cameras are used to create a 3D representation of the environment and the user, usually with the hopes of increasing accuracy or the robustness of the system [3, 35, 45]. While these systems are popular for research or very specific applications, finding a typical computer user with a stereo camera setup is uncommon. For this reason, there has been much research done on

gesture recognition with a single camera as the input [26, 50 51]. The benefit of a single camera system is that cameras are becoming commonplace (many laptop computers have webcams built in now), so the chances are good that a typical user would not have to spend any extra money to use a gesture-based system. Additionally, while recognition with a single camera may not be quite as accurate as a stereo camera setup, it can still be quite good and very usable [51]. While the visual approach to gesture recognition has the advantage of not needing to be attached to the user, the approach can be sensitive to lighting changes or occlusions of the incoming image, whereas sensor-based gesture recognition is obviously free from these issues [25].

2.3 Existing Computer Systems using Gestures for Control

Having discussed the different types of gestures typically used for control, as well as the methods used for capturing those gestures as inputs, a review of some of the existing systems that use these gestural methods of control is warranted.

Figure 1. Individual using the Atlas Gloves application to navigate Google Earth [4].

Freeman et al. created a system whereby a television is controlled using hand gestures [13]. The system employs a single camera to visually identify the user's hand and track it to control a television. Additionally, Merdes et al. created a system called SlidingMap that uses the inclination of a user's hands to control the panning of a map on a tablet PC. The gestures are captured by a dual axis accelerometer embedded in the tablet PC [33]. Another system that uses hand gestures is a project called Atlas Gloves. Atlas Gloves is a hand and arm gesture interface for 3D mapping applications like Google Earth. The system works by using a single camera to identify the hands of the user and capture the gestures as input to navigate within a 3D mapping application [4]. Figure 1 shows the Atlas Gloves project in use.

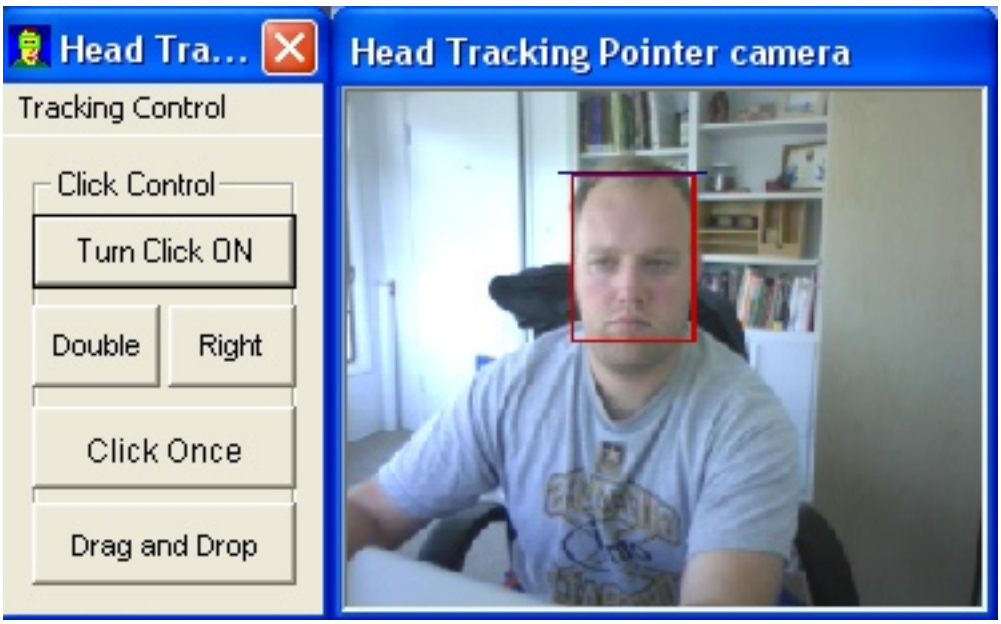

Figure 2. The GUI portion of the Head Tracking Pointer application developed by Kjeldsen in [25]. While this application is running, the computer cursor is controlled by head movements.

While the previous systems all use hand gestures for control, there are several systems that have been developed that use other types of gestures. Sparacino et al. developed a system that utilizes hand and head gestures as a mode of control for navigating in the 3D representation of the internet [48]. For this system, a stereo camera apparatus is used to identify the user's hands and head and track them to capture the user's gestures. One type of gesture that is not used as often is the facial expression. Del Valle et al. developed a system that tracks head pose and facial expressions to control an avatar on a video conferencing application [8]. The recognition and tracking for the system is done visually using a single camera. The last system to be mentioned was developed by Kjeldsen and it is called the Head Tracking Pointer [25]. It uses a single camera to track the motions of the user's head to control a cursor on the screen. Figure 2 is a screenshot of the Head Tracking Pointer application that The Head Tracking Pointer uses a very similar approach to the approach used in this thesis.

2.4 Algorithms for Visual Recognition and Tracking of Gestures

Gesture Recognition algorithms can be broken into two types, object recognition and motion tracking [26]. In some cases, the same algorithm can be used for both recognition and tracking, but this is not always the case. In this research, a simple training step is executed by the user, thereby removing the need for facial recognition. For this reason, only motion tracking algorithms that are being used in different gesture recognition systems are presented here. Section 3.4 is a discussion of the criteria used to determine which of the following tracking methods to use in this research.

2.4.1 Hidden Markov Models

Hidden Markov Models (HMM) are a dynamic programming technique that can be used for pattern recognition or forecasting tasks [36]. What differentiates a Hidden Markov Model from a more basic Markov Chain is that the underlying model is hidden from direct observation, but there is an output model that is dependent upon the hidden model. By using knowledge about the probabilities of an output and knowledge about the probabilities of a state to transition to a different state, information about the underlying model can be inferred [44].

A single HMM consists of a collection of possible states, a transition probability matrix that describes the probabilities of one state transitioning to another state, and finally either an output probability matrix or a continuous output probability density function [43]. The output matrix or function defines the probability of each output given the current state of the model.

Three problems must be solved to use the HMM for pattern recognition or gesture recognition: the learning problem, the evaluation problem, and the decoding problem. The learning problem is solved to train the HMM, the evaluation problem is solved to identify discrete gestures, and the decoding problem can be solved to identify continuous gestures [43].

The general process for setting up a system to recognize gestures using HMMs is as follows: Define the gesture vocabulary to be recognized. Describe each gesture as an HMM, with one HMM per gesture to recognize. This means defining the structure of the

HMM. That is to say, defining how many states and how many values in the various probability matrices are going to be used. The values within these state and probability matrices are not calculated until the training process occurs. Once the training data is collected and preprocessed into a concise and invariant form, the data is used to adjust the model parameters to maximize the probability within the model for the specific gesture being recognized. This adjustment can be done using the Forward algorithm or the Baum-Welch algorithm, a discussion for both of which can be found in [44]. Once the training is complete, gestures can be evaluated against the different models using the Forward-Backward algorithm or the Viterbi algorithm to recognize individual or discrete gestures. Additionally, at this point the Viterbi algorithm can be used as a solution to the decoding problem to identify continuous gestures. A discussion of the Forward-Backward and Viterbi algorithms can be found in [44].

2.4.2 Kalman Filtering

Kalman filtering uses information about the current state of some system, a linear model of behavior, and an element of Gaussian noise to estimate the next state of the system [12]. Kalman filtering is a recursive solution. This means that each new estimate of the state is calculated using the previous estimate and the new input data. Consequently, only the previous estimate must be stored reducing the amount of data that both must be stored and that must be used in the computations of the new estimate. This makes Kalman filtering more computationally efficient than using the entire set of previously observed data to calculate the next estimate. In the case of motion tracking,

the filter will predict the position of an object's bounding box within a two dimensional image [28]. In many cases of motion tracking, the time intervals between measurements are small enough (i.e. one measurement per frame, with high frame rates) that velocity is considered constant and acceleration is considered as white noise in modeling the motion of an object. Therefore, the object being tracked is given a position and a velocity. Using the initial state, which is calculated by finding the change in position between two frames, and the equation of motion that is known to describe the motion of objects within a given domain, a prediction can be calculated of the location of an object in the next corresponding state. This helps to reduce the search space for the recognition task. The downside to using Kalman filtering is that it can be quite cumbersome to create and apply a proper model for estimating the behavior of the system and each system must be specifically tailored to the domain [28].

2.4.3 Particle Filtering

The main idea behind a particle filter is the application of a Bayesian filter, based on sample sets of input data, to incoming data [38]. Particle filtering uses random sampling to compare color histograms at certain points using a similarity measure, such as Bhattacharyya distance, and then estimates the point in the image that most closely matches that distance [38]. One advantage to particle filtering is that it requires no model, but as a result, has a higher computational load [12]. This algorithm does better when the underlying model of behavior is not linear and the element of noise is not Gaussian. With particle filtering, an increase in the dimensionality of a problem leads to a significant

increase in computational complexity. Particle filtering can be more accurate than Kalman Filtering, but it comes at a cost to computational complexity. A hybrid approach with an initial step using a Kalman filter to reduce the dimensionality of the problem followed by the use of a particle filter to come up with a final solution can lead to a system that benefits from the lower computational complexity of the Kalman filter with the increased accuracy of the particle filter [55].

2.4.4 Normalized Cross-Correlation

Normalized cross-correlation is a statistical method for identifying a pattern within a larger set [11]. An early use for cross-correlation was in dynamic signal processing, where a signal was being searched for the occurrence of a particular wave form [11]. As it turns out, this method can also be applied to many other areas where pattern recognition is useful, including image processing [11]. In image processing, the registration of a sub-image within a larger image can be calculated using cross-correlation. The basic idea behind cross-correlation is that the similarity (with regards to Euclidean distance) is calculated between the smaller target image and all possible areas of the larger image in which the search is taking place. The formula for this calculation is:

$$\frac{1}{n-1}\sum_{x,y}\frac{\left(f\left(x,y\right)-\bar{f}\right)\left(t\left(x,y\right)-\bar{t}\right)}{\sigma_f\sigma_t}$$

Where $f(x, y)$ is the image data of the sub-image, $t(x, y)$ is the image data of the template, n is the number of pixels in $f(x, y)$ and $t(x, y)$, $\bar{f}$ is the mean of the sub-image data, $\bar{t}$ is the mean of the template data, and σ_f and σ_t are the respective standard deviations of the sub-image and template data.

The sub-image within the larger search image that has the greatest cross-correlation value is the closest match to the template image. The strength of this approach is that the implementation of the calculation is relatively simple. This simplicity does come with a cost. Cross-correlation is both scale, rotation, and perspective dependent, and for this reason is only useful in specific situations [25]. Additionally, if the smaller target image is rather homogenous and the larger search image has many areas with similar colors, the algorithm will not perform very well. On the other hand, in environments that don't change dramatically and have significant contrast, this technique can be useful for tracking a particular sub-image as it moves within a larger image.

2.5 Systems based on Spatial Information

The number of systems that operate on, analyze and display spatial information is increasing at a rapid pace. The primary reason that this is notable is that many of these tools are being created with the casual user in mind, as opposed to a narrow field of experts for which tools like these were designed in the past. Here, these systems are categorized in to four main groups: 2-Dimensional Mapping Applications, 3-Dimensional Mapping Applications, 3-Dimensional Virtual Environments, and Geographical Information Systems (GIS).

2.5.1 2D Mapping Applications

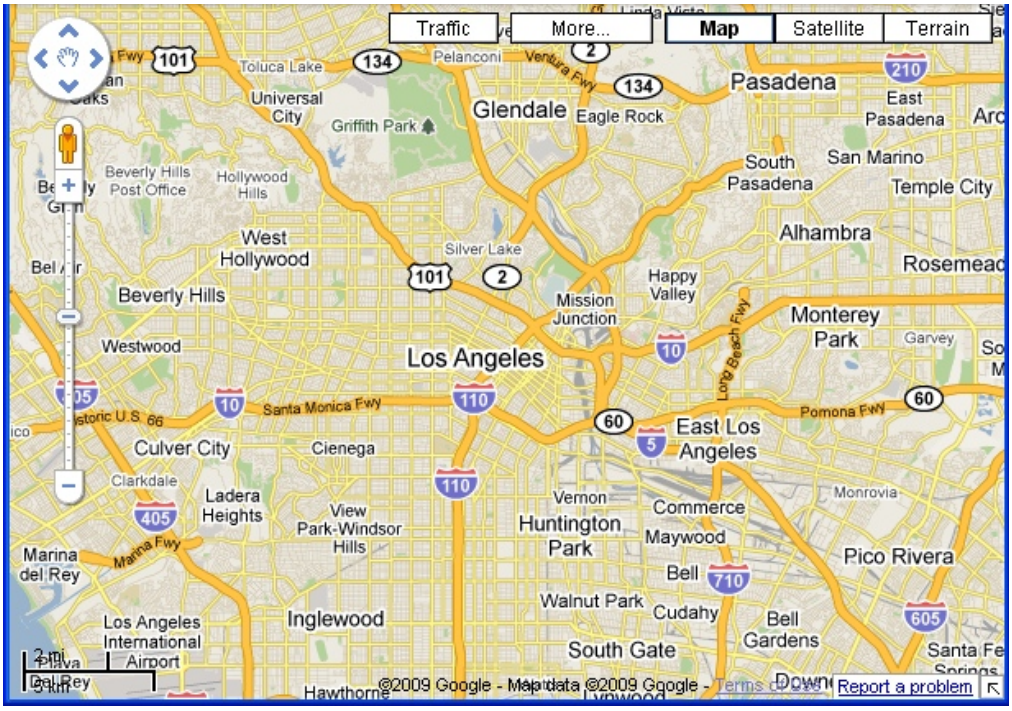

Figure 3. Screen capture of the Google Maps 2D-mapping application. In this screen capture the focus is on Los Angeles and surrounding cities.

2D mapping applications are among the most common emerging spatial information applications. For the general public, this class of application holds the most utility on a day-to-day basis. Applications such as Google Maps from Google, Inc. [17], Yahoo Maps from Yahoo and Mapquest [54], Virtual Earth from Microsoft [34], all can be used by the casual user to find directions to and from user defined locations. One sign of how ubiquitous these applications are becoming is the fact that these mapping applications are being integrated directly in to the largest search engines in use today. 2D mapping of the physical world is not the only type of application that uses spatial relationships though. Another application type that is similar is the idea map, or concept map, which maps some concept space instead of mapping the physical world. Applications such as XMind from XMind, Ltd. use location and proximity to establish relationships between ideas creating a two or three dimensional space to navigate the ideas [53].

2.5.2 3D Mapping Applications

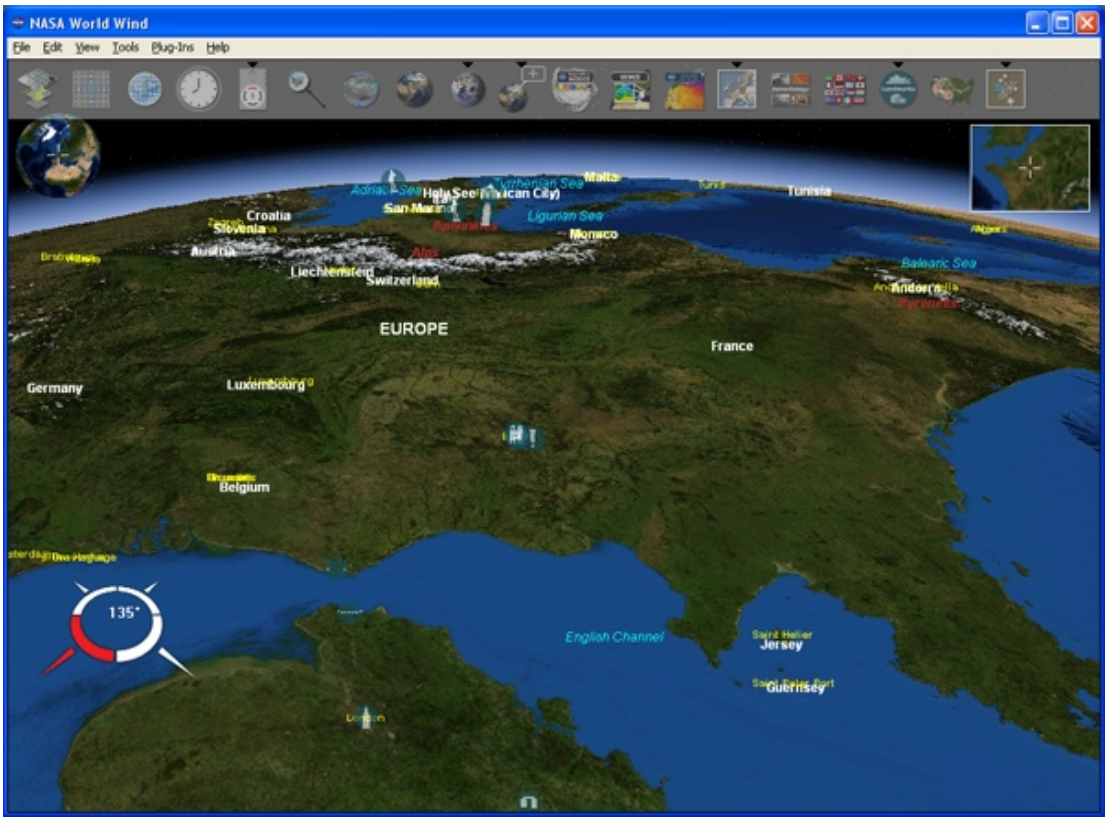

Figure 4. Screen capture of the NASA World Wind 3D-mapping application. In this screenshot several European countries and landmarks are present.

3D mapping applications are a natural extension of 2D mapping applications. By incorporating the 3rd dimension, a more realistic representation of the space being mapped can be created. This can lead the user to gain a more thorough understanding of a space. For a long time, the only systems that could handle 3D mapping were large systems accessible only to large institutions. Over the last decade as computers have become more powerful and data storage and bandwidth have become cheaper, systems like Google Earth [15], Virtual Earth from Microsoft [34], and World Wind from NASA [37] have become available to a more mainstream user base. To show just how

mainstream, as of February 2008, over 350 million people have downloaded Google Earth since it was released [16].

2.5.3 Virtual Environments

Figure 5. Screen capture of the Second Life 3D Virtual Environment. The character is in a fictional place called Help Island.

Virtual environments are another type of application that relies on navigating a space. These are usually not a representation of the real world, but in the case of Second Life from Linden Labs, it is a fictional place for people to meet, play games, chat, buy and sell things, and create user defined places and objects [47]. In Entropia from MindArk, it is a game, but with an economy that allows users to turn in game money into real dollars and vice versa [9]. Kaneva is a world in which people create avatars that can

meet and play games and chat, but it is also a place where companies can create content and use it as advertising [24]. These are all make believe worlds, but the same 3D rules that apply to mapping, apply in these applications as well. So the needs for navigation are the same as in the 3D mapping tools.

2.5.4 Geographic Information Systems

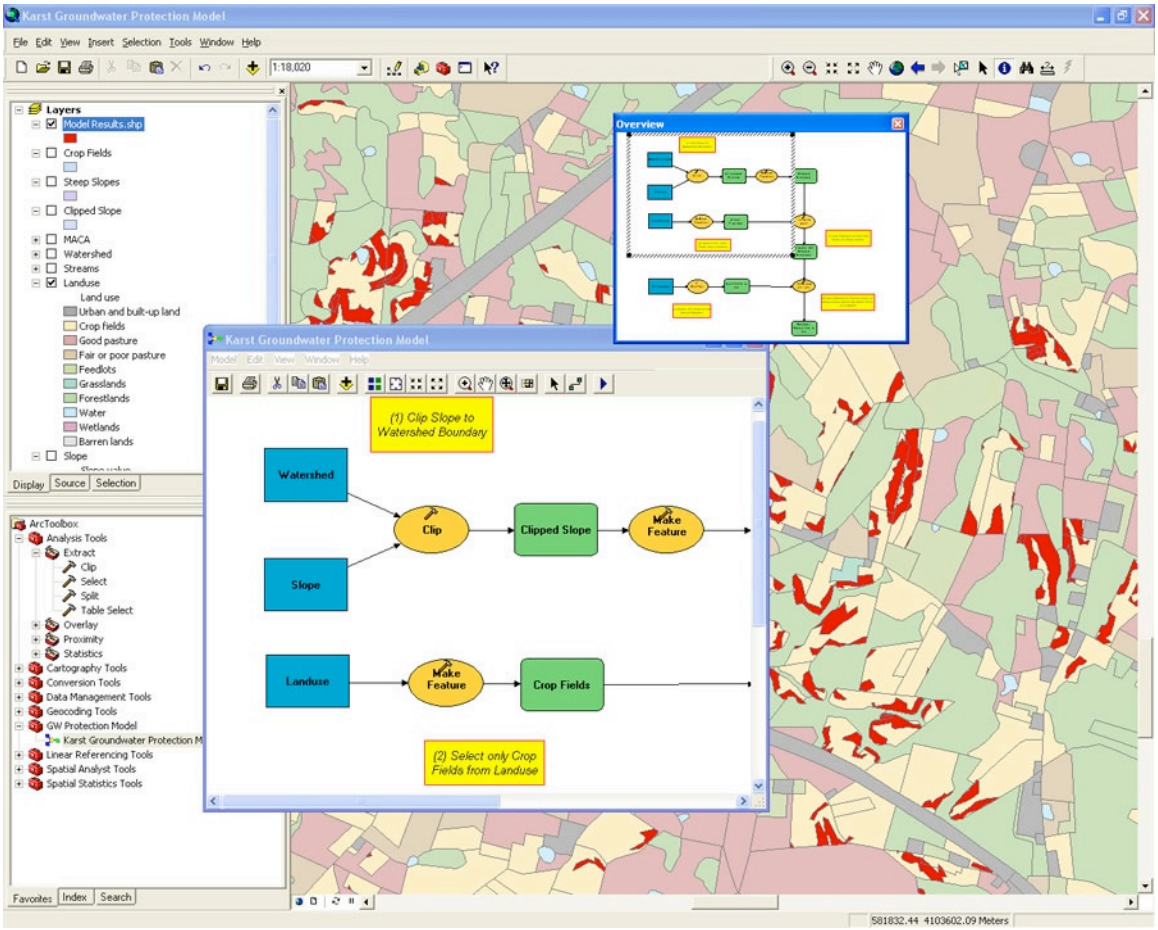

Figure 6. Screen capture of the ArcGIS Geographic Information System from ESRI. In this screen capture, a groundwater protection model is transposed over a geographical region [2].

Geographic Information Systems (GIS) are focused less on casual use and more on using data that has a spatial component for research, emergency response and coordination, planning, and asset management. ArcGIS from ESRI is a system that can be tailored for business, governmental, or educational uses [2]. Information about demographics, or historical information for a given region can be overlaid on a map for which that information is applicable. MapInfo from Pitney Bowes has been used for everything from mapping railways to analyzing crime in major cities to managing water systems to keep them flowing efficiently [32]. Similarly, GeoMedia from Intergraph is targeted towards security, government, and infrastructure projects [14].

3. Design and Implementation

3.1 Inspiration

The design of this application was guided by a particular vision of how a typical user might actually use head gestures for navigating spatial data in the real world, and how it could be made accessible to a large number of users. Many of the studies related to head tracking require very specific hardware as well as a custom, and sometimes laborious, setup, such as the setup seen in Figure 7.

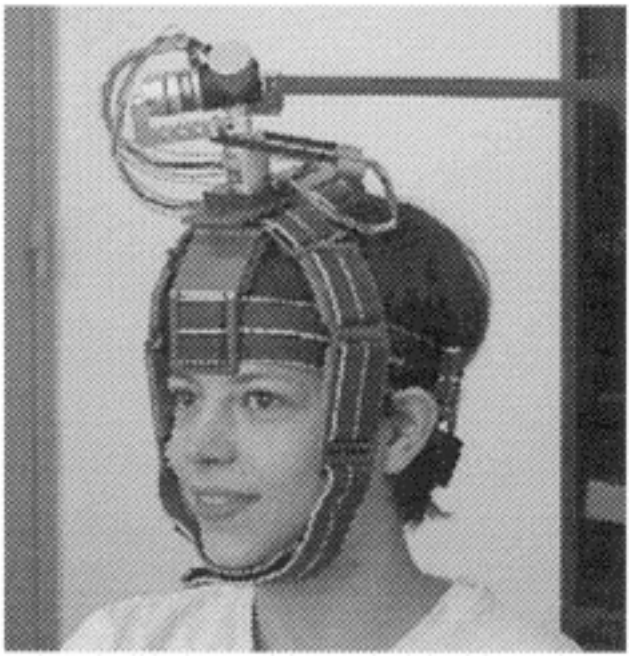

Figure 7. A complicated apparatus for tracking the movements and location of a user's head [40].

For this application, the vision was to have a system that works across platforms and with very little setup required. Obviously, there are certain hardware requirements, such as a camera, but these are becoming more prevalent with the built-in cameras in most new lap-top computers. Because lap-top computers are a major driver for cameras

becoming commonplace, and because the trackpad input method is also being evaluated, many of the design decisions are made with a lap-top computer in mind.

3.2 Key System Requirements

With the previously discussed vision as a guide, a number of decisions were made about the requirements of the system upon which the application would run. The first critical requirement was that the head tracking must be done using a single, common webcam. This was important because they are readily available and don't require significant cost or complicated set up, increasing the number of users to which this application would be accessible. The second requirement was that the system should not require more than average computing power. In this case, average computing power was defined as a system with a 1.7 Ghz Intel Pentium 4 CPU, 1 GigaByte of RAM, and an integrated graphics chipset. Again, this requirement was specified in the spirit of increasing the number of users that could potentially use the system. Finally, the last system level requirement for this application, was that it be operating system independent. This requirement was decided upon, again, to open up the application to as many users as possible. A summary of these system level requirements and the evaluation is included in Table 1.

	REQUIREMENT	EVALUATION CRITERIA
SR-1	Tracking must be completed using a single, common webcam	Application can use a single webcam, either built in to the system or external
SR-2	The application must run on a system with average computing power	The application can run on a system with a 1.7 GHz Intel Pentium 4 CPU, 1 GB of RAM, and an integrated graphics chipset
SR-3	The application must be operating system independent	The application can run on a system running Window, OSX, or Linux

Table 1. System-level requirements and the corresponding evaluation criteria.

3.3 Key Application Requirements

With the system requirements specified such that a large number of users could use the application with their current systems, the application also has a set of requirements to ensure that the evaluation of the head tracking as a method of user input is focused and clear. The requirements are to have the majority of the screen show the map and not be too encumbered by the application itself. The application must provide feedback to the user so that the user can see that the tracking is occurring correctly and also, so they may correct any issues with the training step required for the tracking algorithm. This is especially important because the user is unfamiliar with this method of input and this helps in reducing the learning curve by providing transparency of what the application is seeing and how it is responding. The application must also provide a mechanism to easily set up each evaluation task to streamline the data collection process. In data collection process, users are timed in the completion of a series of tasks and the accuracy of their performances are comparatively ranked between input methods for a given task. For this reason, AR-3 is especially important to keep the user's focus on the tasks at hand, and not on the administration those tasks.

	REQUIREMENT	EVALUATION CRITERIA
AR-1	The majority of the screen must show the map	The application does not take up more space than a small corner of the map
AR-2	The user must receive visual feedback from the application regarding the state of tracking as well as current position on the map	The application provides a video stream from the webcam with an overlay of lines indicating the tracking decisions being made be the application
AR-3	The user must be able to set up a task or reset a task easily	The application provides a button to set up or reset a task with a single click

Table 2. Application-level requirements and their corresponding evaluation criteria.

3.4 Decision Criteria for Choosing the Tracking Algorithm

In deciding which of the four tracking algorithms to use for this application, four criteria were used in evaluating the algorithms. The first criterion was accuracy. For this domain, is the algorithm accurate enough to do the job? The second criterion was speed of execution. Could the algorithm run in real time? The third criterion was ease of training and configuration. Could the algorithm easily be configured for different users? The fourth criterion was ease of implementation. Could the algorithm reasonably be expected to be implemented by a single person in an appropriate amount of time? Table 3 contains the results of the evaluation of these criteria across the four tracking algorithms.

	ACCURACY	SPEED	TRAINING	IMPLEMENTATION
Hidden Markov Model	Very Accurate	Fast	Complex	Must be combined with a recognition algorithm leading to a complex implementation
Kalman Filtering	Accurate	Fast	Complex	Must be combined with a recognition algorithm leading to a complex implementation
Particle Filtering	Very Accurate	Slow	Simple	Simple implementation, but optimizations including combining with Kalman Filter add complexity
Cross-Correlation	Adequate	Moderate	Simple	Simple

Table 3. Tracking algorithms and their evaluation results.

In examining Table 3, Cross-correlation was determined to be the best fit due to the adequate accuracy, fast-enough execution, considerable ease of configuration and implementation.

3.5 Application Design

3.5.1 Interface Design

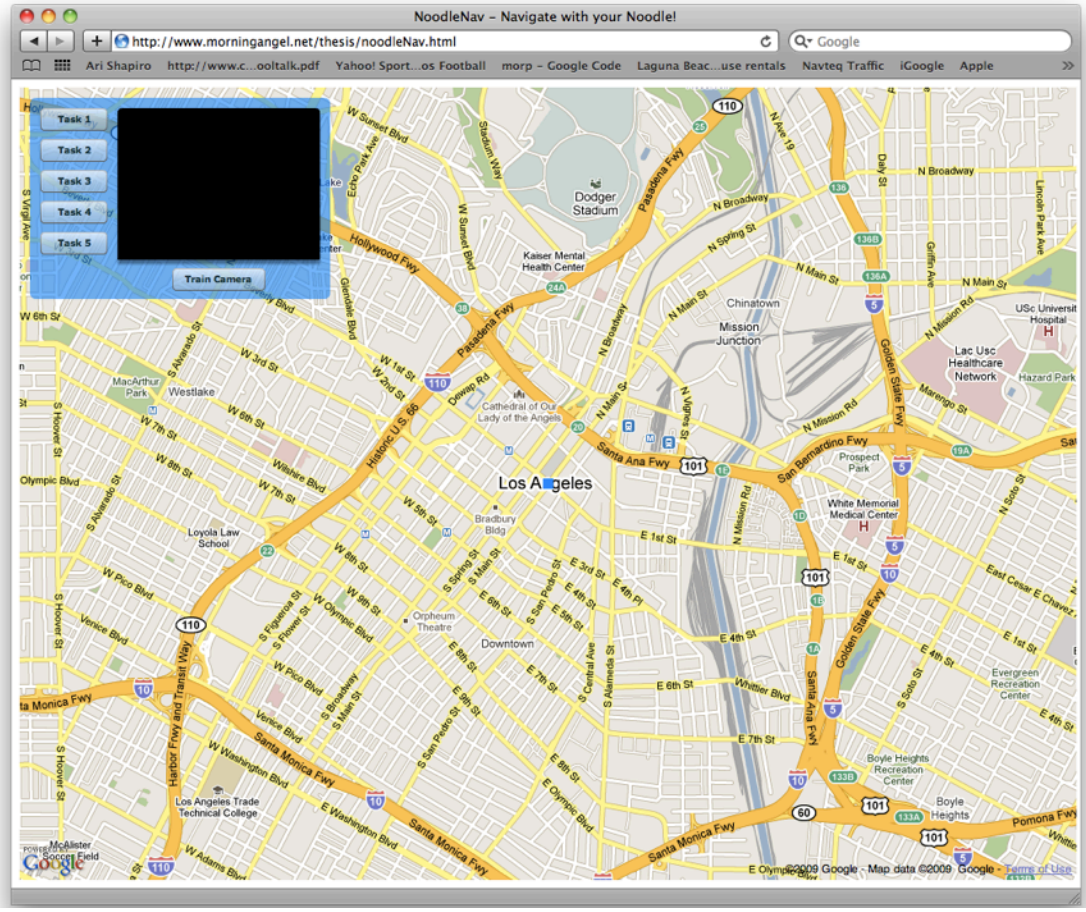

Figure 8. The interface of the application developed for this this. This is the screen the user first sees when going to the appropriate URL.

As seen in Figure 8, the interface is a small application overlay in the upper left corner of a standard Google Map. On the left side of this overlay is a column of buttons corresponding the each evaluation task. When one of these buttons is clicked, it sets the starting and ending flags on the map for that given task. Additionally, it resets the center of the map to the appropriate starting point for the given task.

In the upper right of the application overlay is a panel that shows what the webcam is seeing. Under this panel, there is a button under this webcam panel labeled “Train Camera”. When the user clicks this button, the application then goes in to training mode (Figure 9) where the user is asked to place their head in a predefined box in the webcam panel and click the “Done” button. There is also the option to cancel the training at this point which returns the user to the starting window. Once the user clicks done, the application immediately goes in to tracking mode (Figure 10).

In tracking mode, there is a small blue box that indicates the location of the reference sub-image obtained during the training step, and a small red box that tracks the portion of the current frame’s sub-image that most closely matches the reference sub-image. The difference in location between these two boxes determines the direction and speed that the underlying map pans. In tracking mode, there is a “Stop Tracking” button that stops the tracking and brings the user back to the starting screen.

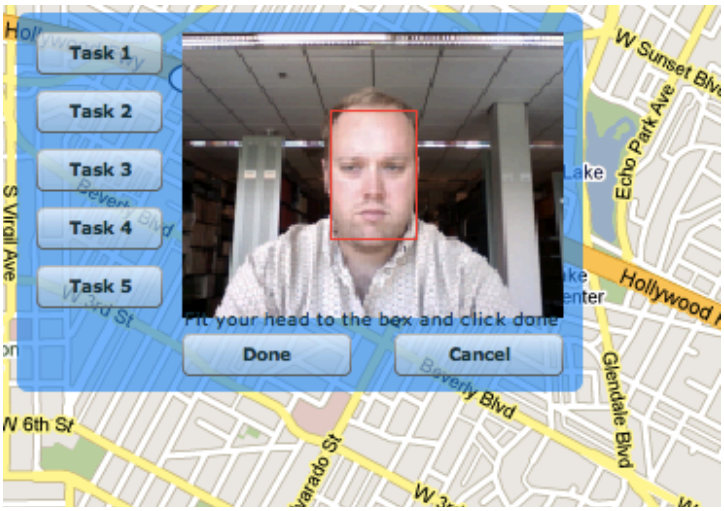

Figure 9. Training panel with instructions for the user as well as buttons to cancel or finish the training step.

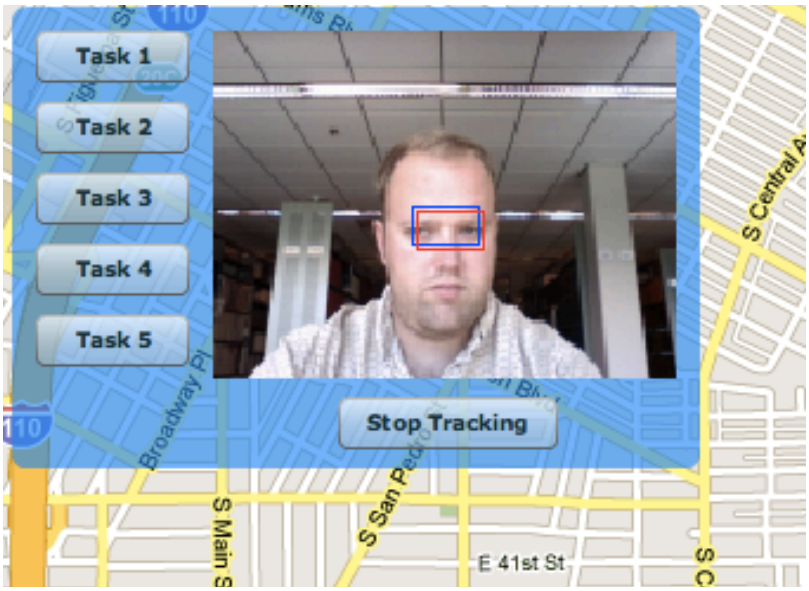

Figure 10. The panel that shows while tracking is occurring. The blue box indicates the position of the template image retrieved from the training step and the red box indicates the current position of the closest match to the template. The difference in these two locations provides a vector to pan the map.

3.5.2 Class Design

Four classes were used to design a solution to fulfill all of the requirements. The main application class, called NoodleNav, is a class that controls the overall layout and behavior of the application. It uses three classes to create the different parts of the application. The class that displays the map on the screen is called GMap and is provided by the Google Maps API. The class that provides the training functionality for the application is called WebcamTrainingPanel. The purpose of this class is to provide the user with instructions for the training step and then capture the reference image for use in the head tracking. Finally, the class that performs the head tracking and map panning is called WebcamPanel. The classes are presented in diagram form in Figure 11.

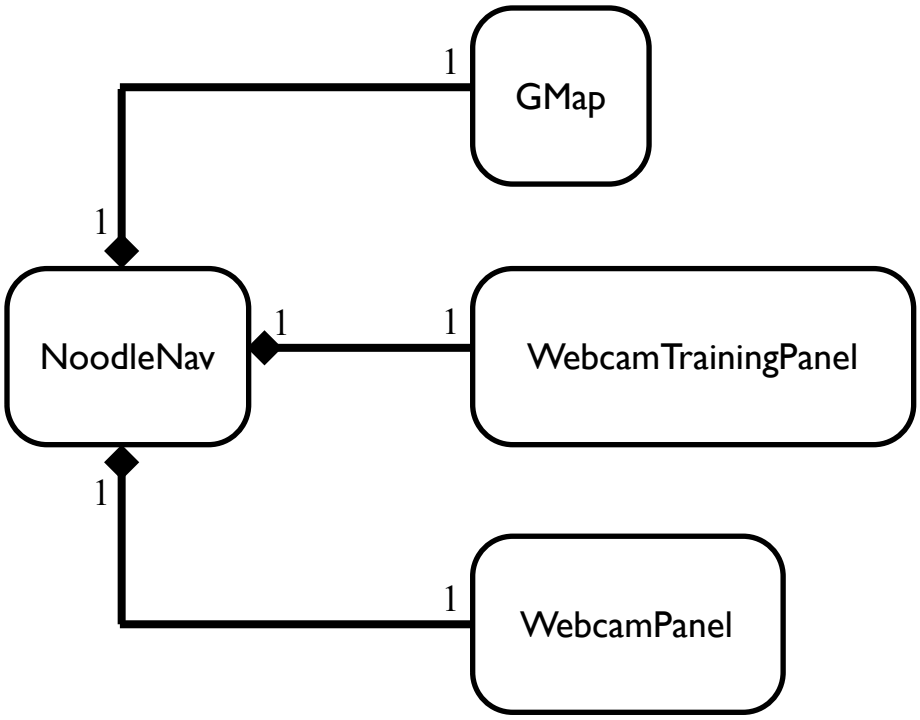

Figure 11. Object diagram for head tracking application.

3.6 Implementation

3.6.1 Development Language and Environment

In order to fulfill the system-level requirement, SR-3, the decision was made to use Adobe Flex 2.0 along with the Google Maps Flash/Actionscript API. This allowed for the fulfillment of the requirement of operating system independence. Additionally, it sped development by taking advantage of the webcam libraries included in the Adobe Flash engine and the existence of the Google Maps Flash API. Finally, the choice to use Google Maps was natural because Google Maps is now the most popular mapping application on the internet, according to internet polling company Hitwise [22], which would increase the likelihood that the users were already somewhat familiar with the mapping environment.

Adobe Flex is an environment used to develop for the Adobe Flash engine. It uses a combination of a markup language (MXML), and a functional language (Actionscript), to specify the look and feel of the application, as well as provide the business logic behind the user-interface. MXML is a markup language that is an extension of XML used to define the layout and behavior of the user-interface components as well as define the transitions between UI components. Actionscript is an object-oriented, functional language that is generally used to provide the functionality to the application. A typical Flex application will have various MXML files that define a user interface which in turn uses Actionscript, either within the same file, or as an instance of some class, to implement the required functionality. Once compiled, a single file with the extension .swf is produced. A link to this file is imbedded in an HTML (noodleNav.html) file which is distributed by the web server.

3.6.2 Interface

The application is implemented using one MXML file (noodleNav.mxml) to define the different aspects of the user interface while instantiating two different Actionscript classes (WebcamTrainingPanel and WebcamPanel) to handle the training task and the head tracking/map panning respectively. Initially, there was some confusion from users because the panel was not acting like a mirror, but instead showing things exactly as the camera was seeing them, so when the user looked left, in the panel it looked as if they were looking to the right. Once this was changed to act more like a mirror, it became much more user friendly.

3.6.3 Algorithms

3.6.3.1 Training Algorithm

Because the training step is not the primary focus of this research, the algorithm chosen for training is very simple. It requires the user to handle the placement of their head within the webcam frame and then assumptions about head position, size, and facial structures are made to estimate the location of the area of the face around the eyes and nose. This meant that a certain spot in the frame is always used as the reference sub-image. The eyes and nose are important, because this area provides enough detail and contrast within an image to differentiate it from other elements of the face. The logic for this is implemented in the Actionscript class called WebcamTrainingPanel. This training step is an area that could be improved. Some ideas for improvement are presented in the future work section of this paper.

3.6.3.2 Tracking Algorithm

The algorithm used to complete the head tracking task is an implementation of cross-correlation. At a regular interval of 75 ms, the frame from the image stream coming from the webcam is processed using cross-correlation. The calculation of the cross-correlation is done with the reference image being the sub-image obtained during the training step and the search space is a larger sub-image of the frame captured from the webcam. The size and location of the search space is based on the location of the previous results of the cross-correlation calculation. By realizing that a user's head is

only going to move so far from frame to frame, the search space can be constrained to an area just around the previously calculated result of the cross-correlation. This improves performance of the tracking considerably by reducing the area to be searched. Once the search space is established, the calculated sub-image with the highest cross-correlation value in the search space (calculated using the formula described previously in the related work section), is the best match to the reference sub-image. Using this technique, the area in the image that is the closest match to the sub-image will be updated and consequently tracked at a rate of about 14 times per second. This rate was empirically determined to provide a sufficiently smooth experience while also providing adequate performance on many different systems. Cross-correlation was chosen for several reasons, the first of which was that cross-correlation is relatively easy to implement and achieve adequate performance with images of this size. If the images being used were higher resolution, cross-correlation might not be fast enough to provide smooth tracking on an average system. Another reason cross-correlation was chosen was due to the nature of the webcam and head tracking, in that the image is fairly static (i.e. the differences from frame to frame are often small), because the movements are often small head turns. For this specific type of situation, cross-correlation works well. For this system, the cross-correlation tracking algorithm was implemented in the WebcamPanel Actionscript class.

3.6.3.3 Translating Tracking into Movement of the Map

The translation of the user's movements into movements of the map is relatively straightforward. The training step provided the anchored location of the original screen capture. The position of the subsequent sub-images that are retrieved during tracking are compared to the position of the sub-image captured during the training step. The difference in these positions gives a vector of direction and magnitude which is then used to make a call to the PanBy method of the GMap object provided by the Google Maps API. This, in turn, leads to the map panning by the appropriate amount and in the appropriate direction to match the user's movements.

4. Evaluation Methodology

4.1 Overall goal of evaluation

The primary goal behind the methodology of this evaluation was to determine how well the head tracking performs the task of panning a two-dimensional map, compared to using a mouse or touchpad as the primary input. To this end, the total evaluation task consisted of a short description of the purpose of the experiment, as well as a few instructions on the use of the different input methods, how the application works, and how the evaluation would proceed. This was followed by a 10 minute period for the user to familiarize themselves with the head tracking, both in training the system and in its use. This period was followed by the completion of five tasks using the mouse, the trackpad, and the head tracking.

Since every user tested has had some experience with a computer, the tasks chosen increased in complexity to act as a sort of tutorial for the head tracking. In the first three tasks, where multiple locations in differing directions were used, the locations were chosen so that the distance from the starting location to the ending location was approximately the same. The first task was to navigate from a starting point in downtown Los Angeles, due west stopping at the Hill Crest Country Club, or due east stopping at Whittier Narrows Golf Course. The second task was to navigate from downtown Los Angeles, due north or due south, to Glendale or South Gate respectively. The third task was to navigate from the starting point in downtown Los Angeles northeast to Pasadena, northwest to Universal Studios, or southwest to Inglewood. The fourth task was to navigate from the starting point in downtown Los Angeles to San Pedro, following the

110 freeway south as closely as possible, while still trying to complete the task in a timely manner. Finally, the fifth task was to start from the intersection of the 405 and 110 freeways, follow the 405 freeway northwest to the 105 freeway, then follow the 105 freeway east to the 110 freeway, then follow the 110 freeway south back to the starting point, completing a full loop. A summary of these tasks can be found in Table 4. After these five tasks were completed, the last part of the evaluation was a free form verbal feedback period, where the user could talk about their thoughts related to the project.

	TASK DESCRIPTION
Task 1	Navigate from downtown Los Angeles, due east or due west, to the Whittier Narrows Golf Course or the Hill Crest Country Club, respectively
Task 2	Navigate from downtown Los Angeles, due north or due south, to Glendale or South Gate, respectively
Task 3	Navigate from downtown Los Angeles, northeast to Pasadena, northwest to Universal Studios, or southwest to Inglewood
Task 4	Navigate from downtown Los Angeles to San Pedro, following the 110 freeway south, as closely as possible
Task 5	Navigate from the intersection of the 405 and 110 freeways, along the 405 freeway to the 105 freeway. Then follow the 105 freeway east to the 110 freeway. Follow the 110 freeway south, back to the intersection of the 405 and 110 freeways.

Table 4. Panning tasks used in the evaluation of the application

To effectively evaluate the performance of the tool, two important characteristics were measured. The first was the objective measurement of time to complete a given task. The second data collected was a subjective ranking of accuracy during the task. This ranking was assigned by the test proctor based on observations of how closely the user was able to follow the directions of the given task and it is a ranking of either 1, 2, or 3 relative to the other modes of input. That is to say, if the mouse was more accurate than the head tracking, but the head tracking was more accurate than the touchpad, the

accuracy ranking would be as follows: Mouse - 1, Tracking - 2, and Touchpad - 3. In order to keep the rankings consistent, the author proctored each evaluation.

4.2 Evaluation Plan Specifics

The sample of users needed to contain a wide range of ages and prior experience with a computer system. This was necessary to help determine both how easily someone with considerable experience with a computer, as well as someone with relatively little experience with a computer, could learn to use the application. To that end, 20 users were evaluated on the system according to the plan described above. The 20 users' ages ranged from 14 to 62, with an average age of 34, and they all had at least some prior experience with a computer. Their self-described experience with computers ranged from two to five on a one to five scale (with one being no experience and five being an expert who used computers on a daily basis in many different ways) and the users had an average experience level of 3.65 on that same scale.

Each evaluation was completed on the same Dell XPS 15-inch laptop, using the same wireless mouse and built-in webcam. For each user, the time to complete the entire evaluation task was approximately 30 minutes.

5. Results

5.1 Analytical approach

In order to ascertain if the data was statistically significant, several statistical techniques were employed in the analysis of the data. In this section those techniques will be addressed along with the tools used to complete the analysis. All statistical calculations were completed on a Macintosh MacBook Pro with OS X. In addition, the Apple Numbers application was used for tabulation and the simple statistical calculations such as Mean, Standard Deviation, and Standard Error. Numbers was also used to create all of the graphs and charts. For the more sophisticated statistical calculations, such as the one-way, within-subject ANOVA and the T-Test, a program called ezANOVA was used. ezANOVA can be used for free and is available at [10].

5.1.1 Mean, Standard Deviation, Standard Error

In completing the analysis of the data, the first step was a straight-forward check using simple statistical calculations to get an idea of what message the data was conveying. This was determined using a calculation of the between-subject mean time to complete the five different tasks for each method input. The mean values provided a simple comparison to evaluate the performance of each particular input method with respect to the other methods. The higher the mean, the longer the task took on average to complete. This alone is not sufficient to conclude that one input method is better than another because mean averages are influenced very heavily by outlier data, particularly with a small sample size such as the sample size used in this research. Standard

Deviation and Standard Error can give some hints as to how well the data actually matches the mean, but even these are not enough to make any conclusions. A stronger statistical test is needed.

5.1.2 One-way, Within-Subject ANOVA/T-Test

The stronger statistical test used to analyze this data is the one-way, within-subject analysis of variance (ANOVA). The result of the calculation provides a probability measurement that the null hypothesis is true [18]. That is to say, it is the probability that the data occurred purely by chance and was not affected at all by experimental manipulations. This probability value is often referred to as the "p-value". It is commonly accepted that a p-value less than 0.05 means that the data is statistically significant [18]. That number means that there is less than a five percent chance that the Null Hypothesis is true, or that the data occurred purely by chance. Statistical significance means that there is a very high probability that it was the experimental manipulations that caused the observed results and not chance.

An ANOVA is used because within the one factor, the input method, there are three levels that are being compared. These levels are the results of the mouse, touchpad, and head tracking respectively. A T-test is another statistical measure that can be used to test the hypothesis by determining the probability that the null hypothesis is true [18]. Unfortunately, a T-test is insufficient because it only compares two levels at a time, e.g. mouse vs. touchpad, which can be hard to interpret for meaning. One the other hand, an ANOVA can compare all three together and provide a p-value for the entire data set.

Once significance is established for each task with the ANOVA, T-tests are used for pairwise comparisons to do individual comparisons between input methods to determine which comparisons were statistically significant, which approached significance, and which were not significant at all.

One other item of note, is that an ANOVA assumes that the data represents a normal distribution. The ANOVA method is quite robust to violations of this assumption [18], but in the case that the data are too far out of normal, certain data transformations can be applied to fit the data in to a more normal distribution [18]. If these data transformations are insufficient, non-parametric calculations can be used, which do not require the data to be normal, but can be harder to calculate and interpret [18]. The data collected for this thesis was also analyzed with the non-parametric Kruskal-Wallis ANOVA, and there was not a significant difference between the values of the one-way, within-subject ANOVA and the Kruskal-Wallis ANOVA. Due to this similarity and to ease interpretation of the results, only the one-way, within-subject ANOVA is presented.

5.2 Results

For each task a bar graph with the Average (Mean) Time for completion, the Average (Mean) Accuracy Rack and standard error for both are presented. The graph contains the data for each of the input methods for the given task. In addition, a table with the ANOVA calculated p-value for the entire task and the T-test calculated p-values for the pairwise comparisons are presented. Finally, these values were calculated over the entire dataset, combining all five tasks into one dataset, and that data is presented as well.

5.2.1 Task 1 - East and West Panning

In task 1, which is focused on east and west panning only, users were fastest and most accurate with the Mouse with an average completion time of 6.88 seconds and an average accuracy rank of 1.65, where 1 is the most accurate and 3 is the least accurate. The second fastest as well as the second most accurate was the head tracking with an average time of completion of 8.41 seconds and an average accuracy rank of 1.70. In this task, the accuracy rank of the head tracking was very close to that of the mouse. Finally, the touchpad was slowest with an average time of completion of 9.80 seconds, and an average accuracy rank of 2.65. This can be seen in Figure 12.

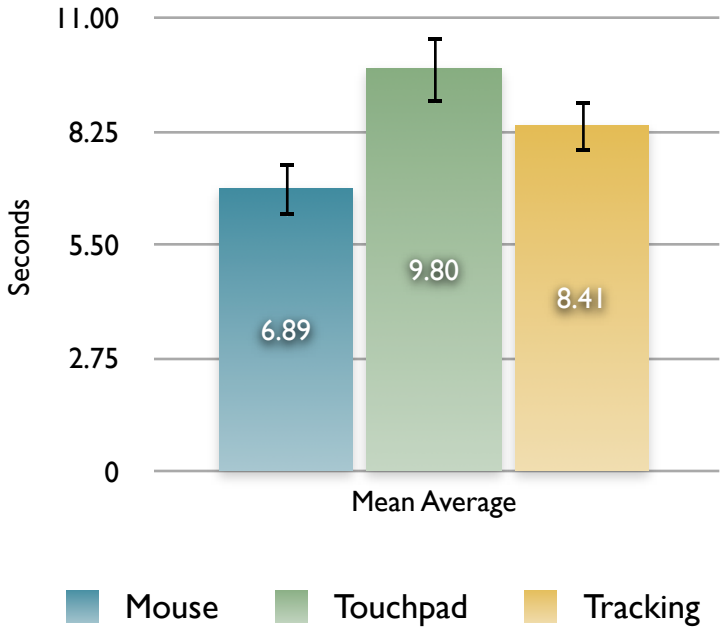

Figure 12. Average Completion Times for Task 1.

One interesting result to note is that, after normalizing the mean averages of the accuracy and time values so they can be plotted on the same chart, (see Figure 13), a relationship can be seen. The faster devices were more accurate and the slower device was less accurate.

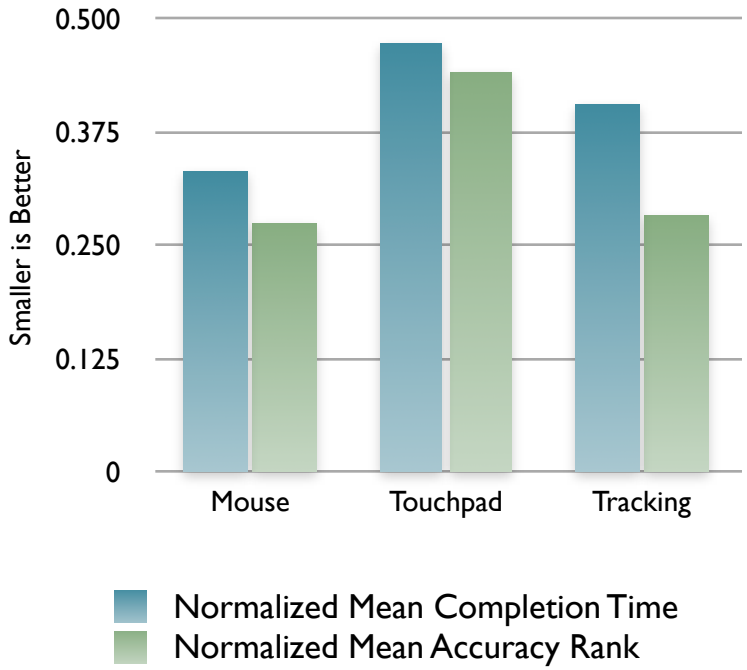

Figure 13. Normalized Mean Completion Time and Normalized Mean Accuracy Rank for task 1.

Finally, in performing the ANOVA on the data for task 1, the overall p-value indicates that the data is indeed statistically significant with a value $p<0.000529$. In looking at the p-values of the pairwise comparisons, the touchpad versus the head tracking comparison approaches significance, but falls just short with $p<0.0893$. The mouse versus touchpad comparison and the mouse versus head tracking comparison are both statistically significant with p-values of $p<0.005$ and $p<0.0108$, respectively. That is

to say that the there was approximately one percent chance that the results occurred purely due to chance. Table 5 shows the results of the ANOVA analysis.

	P-VALUE	SIGNIFICANCE	METHOD
Overall	< 0.000529	Significant	ANOVA
Mouse vs. Touchpad	< 0.0005	Significant	T-Test
Mouse vs. Tracking	< 0.0108	Significant	T-Test
Touchpad vs. Tracking	< 0.0893	Approaching Significance	T-Test

Table 5. P-values for task 1 data, both overall and for each pairwise comparison.

5.2.2 Task 2 - North and South Panning

Task 2 was focused solely on north and south panning motions. In this task, users were again fastest and most accurate using the mouse, with an average time of completion of 5.46 seconds and an average accuracy ranking of 1.30. The head tracking input method was second fastest with an average time of completion of 9.50 seconds and an average accuracy ranking of 1.95. The slowest input method was again the touchpad with an average time of completion of 12.24 seconds and an average accuracy ranking of 2.75. The average times of completion for all three input methods are presented in Figure 14.

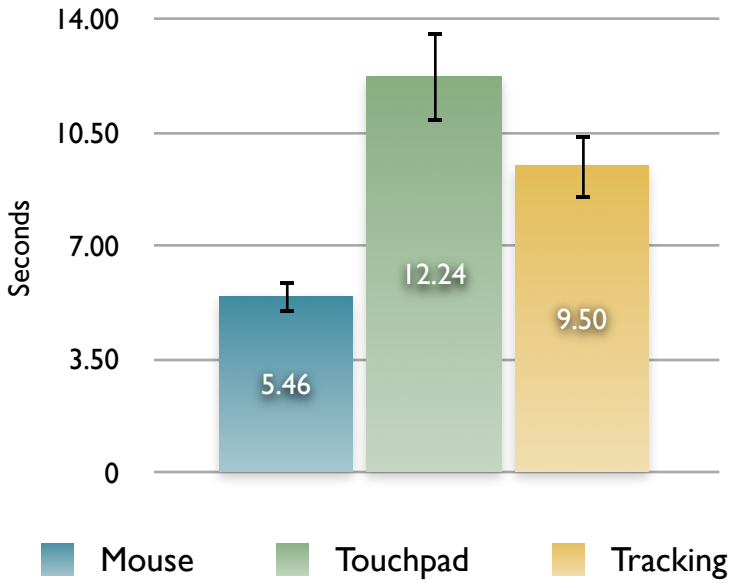

Figure 14. Average time to complete task 2 for all three input methods.

When looking at the normalized average time to complete the task alongside the normalized average accuracy rank (Figure 15), the plots look very similar to the table from task 1, showing that the faster input methods were also more accurate for task 2.

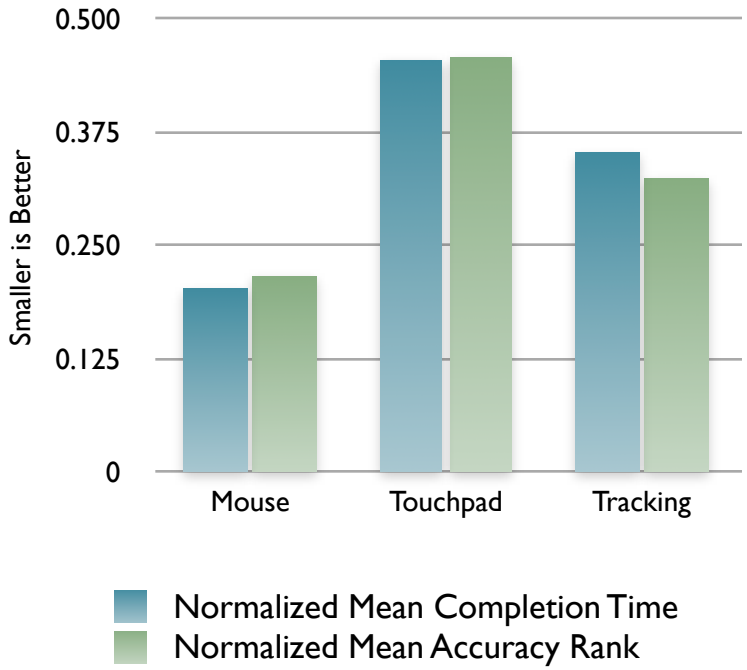

Figure 15. Normalized Mean Completion Time and Normalized Mean Accuracy Rank for task 2.

As with task 1, the p-values for task 2 show statistical significance with an overall $p < 0.000006$ calculated using ANOVA. In the pairwise comparisons, there is a similar situation to task 1. The p-value of the mouse versus the touchpad is a statistically significant $p < 0.0001$. The p-value of the mouse versus the tracking input method also is a statistically significant $p < 0.0001$. The p-value for the touchpad versus the tracking is $p < 0.0804$ which is approaching significance, but actually a bit short. These values can be seen in Table 6.

	P-VALUE	SIGNIFICANCE	METHOD
Overall	< 0.000006	Significant	ANOVA
Mouse vs. Touchpad	< 0.0001	Significant	T-Test
Mouse vs. Tracking	< 0.0001	Significant	T-Test
Touchpad vs. Tracking	< 0.0804	Approaching Significance	T-Test

Table 6. P-values for task 2 data, both overall and for each pairwise comparison.

5.2.3 Task 3 - Northeast, Northwest, and Southwest Panning

For task 3, users were asked to pan to points on the map that were northeast, northwest, and southwest from the starting point in Los Angeles. In this task, users were again fastest panning using the mouse, with an average task completion time of 4.93 seconds and an average accuracy rank of 1.60. Using the head tracking, the users had an average completion time of 9.23 seconds and an average accuracy rank of 1.65, again besting the touchpad, which had an average completion time of 11.11 seconds and average accuracy rank of 2.75. The average completion times for task 3 are presented in Figure 16.

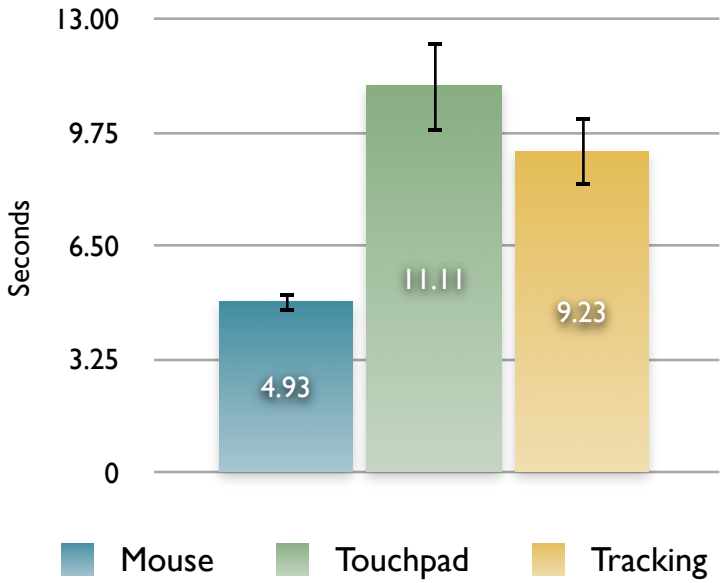

Figure 16. Average time to complete task 3 for all three input methods.

In plotting the normalized average time to completion next to the normalized average accuracy rank, we see again that the faster methods of input were also the more accurate methods of input. In this case though, despite the fact that the mouse was quite a bit faster, the head tracking was nearly as accurate. This plot can be seen in Figure 17.

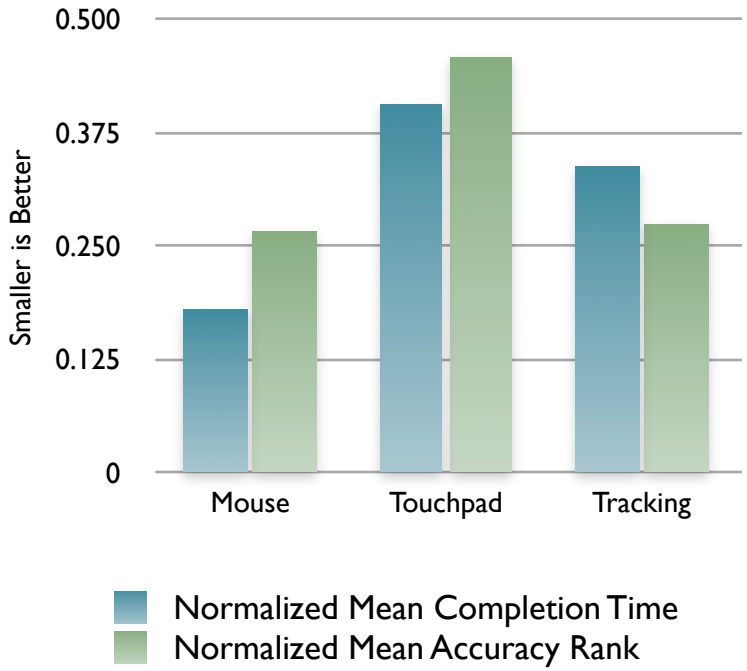

Figure 17. Normalized Mean Completion Time and Normalized Mean Accuracy Rank for task 3.

In task 3, the overall significance of the results, as calculated using ANOVA, showed statistical significance with $p < 0.0006$. Looking at the individual pairwise comparisons, the p-value of the mouse versus touchpad is significant at $p < 0.001$. The p-value of the mouse versus tracking is also significant at $p < 0.001$. In looking at the variability between the touchpad versus the tracking, there is no significance in the result with $p < 0.2605$. These results can be found in Table 7.

	P-VALUE	SIGNIFICANCE	METHOD
Overall	< 0.0006	Significant	ANOVA
Mouse vs. Touchpad	< 0.0001	Significant	T-Test
Mouse vs. Tracking	< 0.0001	Significant	T-Test
Touchpad vs. Tracking	< 0.2605	Not Significant	T-Test

Table 7. P-values for task 3 data, both overall and for each pairwise comparison.

5.2.4 Task 4 - Following a Long Path in One Main Direction

In task 4, users were asked to follow a highway a considerable distance. In this task, the tracking input method is the fastest with an average time of completion of 15.06 seconds and an average accuracy rank of 1.90. The mouse was the second fastest with an average time of completion of 17.68 seconds and an average accuracy rank of 1.55. Finally, the touchpad was slowest, with an average time of completion of 25.01 seconds and an average accuracy rank of 2.55.

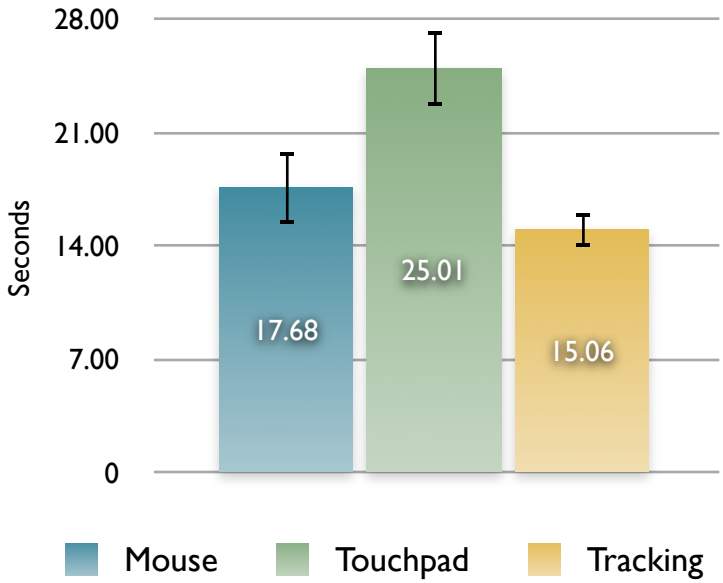

Figure 18. Average time to complete task 4 for all three input methods.

Plotting the normalized averages of the time of completion and the average accuracy rank on the same chart, there is a little difference in this task compared to the previous three. For the first time the fastest input method is not the most accurate. In task 4, the fastest input method is the tracking, but the most accurate method is the mouse. Again, the touchpad is the least accurate of the three input methods. These results can be seen in Figure 19.

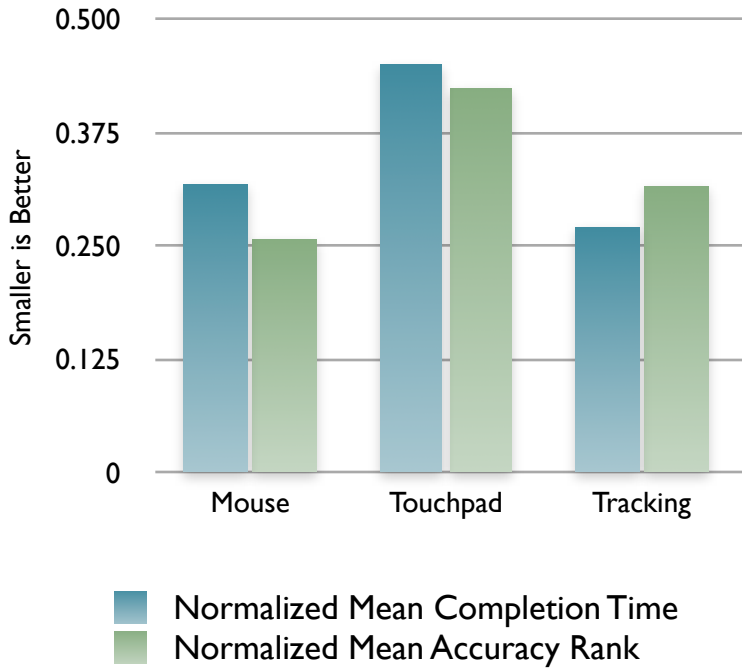

Figure 19. Normalized Mean Completion Time and Normalized Mean Accuracy Rank for task 4.

The overall differences in the average completion times for task 4 proved to be statistically significant with a value of $p < 0.000001$. In comparing the variance of the data for the mouse versus the touchpad, it proved to be significant with a value of $p < 0.0001$. In comparing the variance between the mouse and the tracking, there was no significance with $p < 0.1337$. Finally, the variance between the touchpad and the tracking input, there is again statistical significance with $p < 0.0001$. Table 8 has all of the calculated p-values.

	P-VALUE	SIGNIFICANCE	METHOD
Overall	< 0.000001	Significant	ANOVA
Mouse vs. Touchpad	< 0.0001	Significant	T-Test
Mouse vs. Tracking	< 0.1337	Not Significant	T-Test
Touchpad vs. Tracking	< 0.0001	Significant	T-Test

Table 8. P-values for task 4 data, both overall and for each pairwise comparison.

5.2.5 Task 5 - Following a Circular Route Covering Several Directions

For task 5, users were asked to follow a series of freeways which formed a loop, so that the starting point was also the ending point. In this task, the mouse was again the fastest method of input with an average completion time of 15.43 seconds and average accuracy rank of 1.85. The tracking input method was not much slower at 18.17 seconds, and it had an average accuracy rank of 1.70. Finally, the touchpad was significantly slower than the other two, with an average completion time of 28.09 seconds and an average accuracy rank of 2.45. Figure 20 shows a comparison of average completion times between the three input methods.

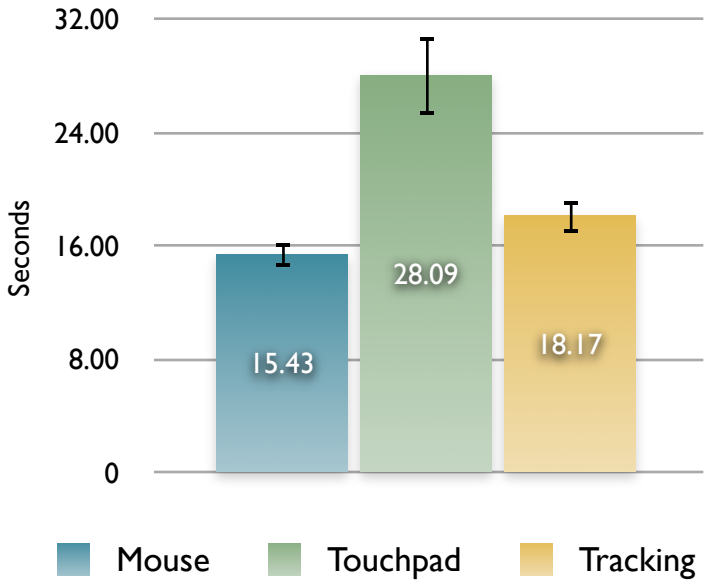

Figure 20. Average time to complete task 5 for all three input methods.

In plotting the normalized average completion times alongside the normalized average accuracy rank, the previous observation holds that the input methods that are faster are also more accurate. The mouse and head-tracking are both very close in average completion time and average accuracy rank, so the slight violation of this observation is likely due to simple variation in the data. Figure 21 shows these two data plotted on the same chart.

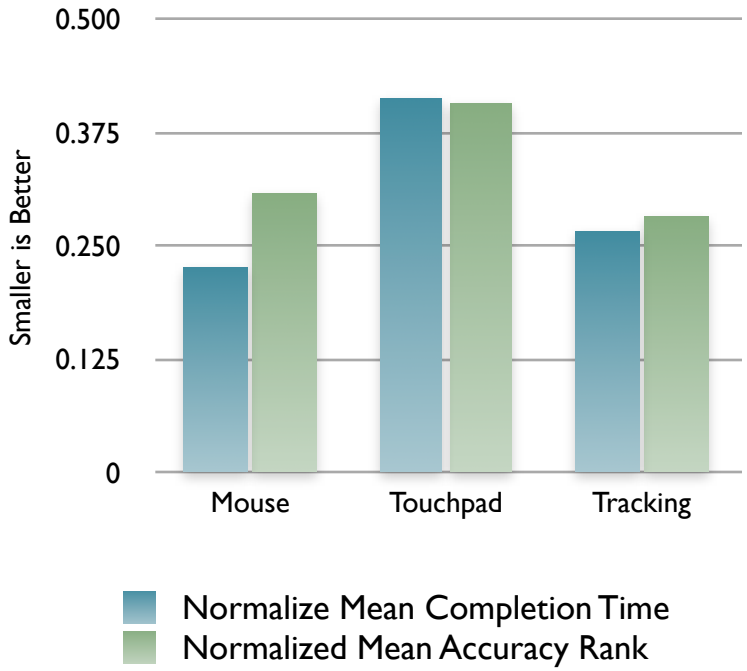

Figure 21. Normalized Mean Completion Time and Normalized Mean Accuracy Rank for task 5.

In task 5, the overall differences in the completion times were statistically significant with $p < 0.000001$. Additionally, all three pairwise comparisons were statistically significant, with the mouse versus touchpad having $p < 0.0001$, the mouse versus head tracking having $p < 0.0266$, and the touchpad versus the head tracking having $p < 0.0005$. Table 9 shows the results of the calculations.

	P-VALUE	SIGNIFICANCE	METHOD
Overall	< 0.000001	Significant	ANOVA
Mouse vs. Touchpad	< 0.0001	Significant	T-Test
Mouse vs. Tracking	< 0.0266	Significant	T-Test
Touchpad vs. Tracking	< 0.0005	Significant	T-Test

Table 9. P-values for task 5 data, both overall and for each pairwise comparison.

5.2.6 Overall Results

The final piece to the puzzle is to look at the overall results in order to assess the performance of the different input methods compared to each other. For this, all results for each task are included in one table and the averages are all calculated. In doing this, the mouse came out on top with an average completion time of 10.07 seconds and an average accuracy rank of 1.59. The head tracking input method was the next fastest, with an average completion time of 12.07 seconds and an average accuracy rank of 1.78. Finally, the touchpad was the slowest, with an average completion time of 17.25 seconds and an average accuracy rank of 2.63. Figure 22 shows the average completion times with the standard error bars.

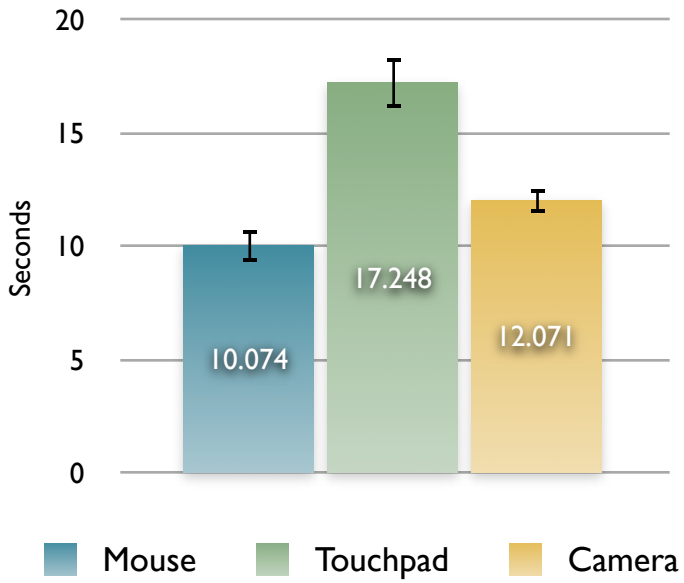

Figure 22. Overall Average completion times with standard error bars for all three input methods.

When comparing the normalized means of the completion times and the accuracy ranks for all tasks, the previous observation holds true in that the faster the method of input, the more accurate as well. See Figure 23 for the normalized average completion time and normalized average accuracy rank plotted on the same chart.

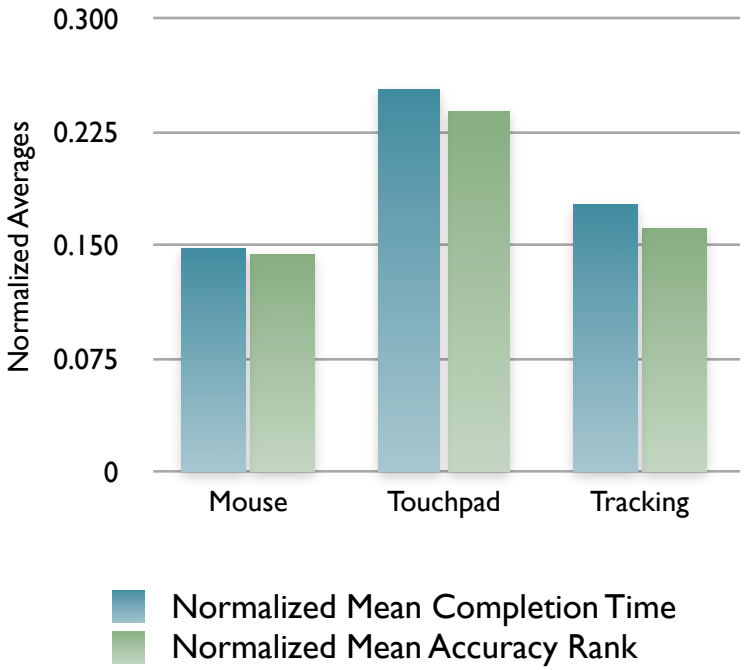

Figure 23. Normalized Mean Completion Time and Normalized Mean Accuracy Rank Overall.

Using a one-way, within-subject ANOVA, the overall set of data is statistically significant with $p < 0.000001$. Looking at the pairwise comparisons, the data of the mouse versus touchpad is significant with $p < 0.0001$. The comparison between the mouse and the head-tracking input method is significant with $p < 0.0003$ and the comparison between the touchpad and the head-tracking input method is significant with $p < 0.0001$. Table 10 has the data and calculation method for the overall dataset.

	P-VALUE	SIGNIFICANCE	METHOD
Overall	< 0.000001	Significant	ANOVA
Mouse vs. Touchpad	< 0.0001	Significant	T-Test
Mouse vs. Tracking	< 0.0003	Significant	T-Test
Touchpad vs. Tracking	< 0.0001	Significant	T-Test

Table 10. P-values for the overall data, both overall and for each pairwise comparison.

5.3 User Comments

After completing the data collection portion of the evaluation, each user was given the opportunity to provide verbal feedback about the tasks they had just completed and the different input methods used. The comments were varied, but after grouping and tallying the comments, there were six different comments that came up 4 times or more. With respect to preferences of which input method they liked best, of the 20 users surveyed, 11 of them specifically mentioned that the mouse was their preferred method of input. 13 users mentioned specifically that they preferred the head tracking to the touchpad, and nine users said they didn't like using the touchpad at all. Of note, no user preferred to use the touchpad over the mouse or head tracking, while three users actually preferred using the head tracking over the mouse. Some users didn't mention any preference.

The rest of the comments that came up most were related to feedback about the head tracking tool. Five users mentioned that they felt with practice they would actually improve further with the head tracking input method, and therefore felt that the evaluation would have done well to allow for more practice. Five users also mentioned

that they would have liked the ability to adjust the sensitivity of the tracking based on personnel preference as some users felt the panning wasn't fast enough and others felt that it was too fast. Three users mentioned that having to maintain a certain posture was a little tiring. Finally, four users pointed out that they would like an easier way of turning the tracking on and off to help manage unintentional panning.

Evaluating the use of the head tracking input method, the general response was positive, with some users saying that it was more intuitive for them to look where they wanted to pan to, as opposed to having to "drag" the underlying map in the opposite direction of the desired direction. In one case, the user had never used a touchpad or the head tracking before the task and by the end that user felt that the learning curve for the head tracking was significantly faster compared to the touchpad. Some users felt that the hardest part of using the head tracking was stopping the panning where they wanted, because they had a tendency to move their eyes and forget to move their heads. Four users had a tendency to look down when they wanted to pan up and look up when they wanted to pan down, but they were fine panning left and right, so they would have liked an option to invert the vertical panning. As far as usefulness was concerned, a couple of users mentioned that they would use it, others mentioned that, while they felt the head tracking was "cool", they couldn't see themselves using it in practice. One user studying to be an occupational therapist mentioned that they could see some clinical uses for the technology.

5.4 Analysis

The results from the previous section indicate that, with respect to panning, the mouse is both the fastest and most accurate input method. Considering that every user had significantly more experience with the mouse, this result is not surprising. Head tracking is about 20 percent slower than the mouse but nearly as accurate. Finally, the touchpad is both the slowest, 70 percent slower than the mouse, and the least accurate. These results prove to be highly statistically significant as calculated using a one-way, within-subject ANOVA for the overall comparison and a t-test for the individual pairwise comparisons. In only a few cases do the pairwise comparisons fail to meet the criteria for significance and in those cases the times are very close to each other. The qualitative responses from the users seem to bear out these quantitative results as generally users preferred the mouse, with occasional preference for the head tracking, while almost universally disliking using the touchpad.

In evaluating the performance of head tracking versus the other two input methods, head tracking performed best in the last two, more complicated, tasks. In these tasks, the users had to pan longer distances than in the first three tasks. In several cases, users were actually fastest and most accurate with the head tracking by this point in the evaluation. It was during these tasks also that most users seemed to respond positively to the head tracking, with one user going so far as to say “Now I get it!” while completing task 4 following the long stretch of freeway.

6. Contribution

In summary, this study shows that with limited exposure and practice, users were able to complete several panning tasks, ranging from simple to complex, using head tracking as a method for input, in less time and more accurately than with the touchpad on a laptop. Users were faster and more accurate with the mouse than with head tracking, but as the tasks progressed, the performance gap between the mouse and the head tracking began to shrink. It is possible that this is a reflection of the amount of experience users had with a mouse versus head tracking, and that given more practice head tracking could equal or surpass the performance of the mouse. The other possibility for this performance gap shrinking is that the head tracking is not as well suited for the simple panning tasks, but in more complex and longer panning tasks, it does better. Further study is required to determine the reason for the increase in performance. The quantitative data correlated with the qualitative data provided by the users, after the evaluation tasks were completed, in that the touchpad was the most difficult and frustrating method of input for panning, the mouse was the most comfortable, and the head tracking was better than they expected and nearly as good as the mouse.

7. Research Validation

Despite the fact that the research was performed with careful organization and planning, there are a few weaknesses that must be addressed. The first weakness is that the tasks were limited to one direction per input per task. So, for task 1 a particular input method either went east or west, but not both. The reason for this was that the evaluation required a significant amount of time, approximately 45 minutes, and to effectively double the number of tasks required was considered too much to ask of volunteer users who were not being compensated. As it was, users began to get fatigued toward the end of each evaluation. It's possible that a between subjects design could be employed to get around this problem, but for the initial evaluation of this concept, it would have had its own problems.

The second weakness with the study is the method of evaluation for the accuracy of each input method for the given tasks. The responsibility for ranking the accuracy fell to the evaluator and was a subjective measure. To minimize the differences in evaluation, the author was the only person to perform any of the evaluations, so that was consistent across all users. A better approach would be to formulate a more objective measure of accuracy that could be calculated either by separate evaluators, or perhaps by the tool itself. One possibility for such a system could be to determine an idealized navigation path and have the application calculate and record the deviation from that path. In doing this, an objective score could be calculated for each input method and each task.

Finally, a big weakness that was not addressed by this research is the differing levels of user experience with the various input methods. The ideal case would be to

have users practice with all three for a significant amount of time, perhaps over several days, and then do the evaluation. Due to time constraints, this was not possible in this research, but it certainly warrants further study.

8. Future Work

There are several areas into which this research could expand. The two main areas for expansion are in the enhancement of the head tracking tool and in the evaluation and test design. The test design has many options for enhancement. An automated way to calculate the accuracy could enhance the accuracy data and might provide some more specific insights on the different ways a user may make a mistake and correction with the different input methods. The idea for another interesting study that could be performed came from one of the users who mentioned that they felt their video game experience helped them perform better with both the head tracking and the mouse. Additionally, one user was a retired fighter pilot who had experience with missile targeting systems that used head gaze for targeting; this user mentioned that the head tracking felt natural. It would be interesting to design an experiment in which head tracking was used to navigate a video game or flight simulator space and compare it to the other methods of input common in that domain. Additionally, repeating the experiment with users who had similar levels of experience with the mouse and the head tracking would be interesting, to see if the head tracking would outperform the mouse or at least be comparable. The hard part of doing that would be finding users with so little experience with the mouse, since it is far and away the dominant method of input for computing. Finally, a simple improvement to the test design would be to create a standard questionnaire for users to fill out after completion of the various tasks as this would be helpful in making the qualitative data easier to compare across users.

Some improvements to the tool itself would be worthwhile towards enhancing performance compared to the other input methods. Since the focus of this research was to evaluate the concept of head tracking as a method of panning spatial data, the algorithm chosen for the actual tracking was a compromise of acceptable performance and easier implementation. Improving the performance of the head tracking, both in different lighting conditions and in response time would be helpful. Additionally, creating a more automated way of training the system, perhaps one that can handle users in different starting positions would go a long way toward enhancing the user perception of the system. Adding in other dimensions of control, such as zooming with a blink, could provide a path for the tool to become more useful and practical for everyday use. Finally, it is hard for the user to get the panning via head tracking to stop exactly where they want it to, so the tracking and response to small movements could use some refinement. Some combination of these enhancements would most likely improve the results of this research.

Extending the tool to explore how well head tracking would work in a 3D environment, particularly in light of the demonstrated ease of use in the 2D environment, would also be a worthwhile endeavor.

9. Conclusion

This research set out to examine and evaluate the performance of head tracking as an input method for panning 2D spatial information. This is an important step toward removing the need for a user to occupy their hands with navigating through information, either to enable disabled users or to enhance the capabilities of a typical user by allowing them to use their hands for other tasks while navigation is completed with natural motions like using head turn for control.

This evaluation was completed in two parts. The first part was to create the head tracking application. This application was created using Adobe Flex, Actionscript, and the Google Maps API. It was created such that it is operating system independent and has a low cost to use, only requiring an internet connection, a standard webcam, and the Adobe Flash player. The second part consisted of 20 users completing five tasks of increasing complexity with three input methods: mouse, laptop touchpad, and the head tracking application. Each task was timed and an accuracy rank was assigned to each input method for each task. The results proved to be statistically significant using a one-way, within-subject ANOVA and revealed that the head-tracking was slightly behind the mouse in performance, but significantly ahead of the touchpad.

This paper concludes that, as a method of input, head tracking provides intuitive and precise control for panning two-dimensional spatial data. With further application refinement and user practice, head tracking may ultimately outperform the mouse in navigating spatial information.

10. References

[1] Agrawal, Pyush, Ingmar Rauschert, Keerati Inochanon, Levent Bolelli, Sven Fuhrmann, Isaac Brewer, Guoray Cai, Alan MacEachren, and Rajeev Sharma. "Multimodal Interface Platform for Geographical Information Systems (GeoMIP) in Crisis Management." *In Proceedings of the 6th International Conference on Multimodal Interfaces.* New York, NY, USA: ACM Press, 2004, 339-340, DOI= http://doi.acm.org/10.1145/1027933.1027997.

[2] ArcGIS from ESRI. Retrieved on November 16, 2009, from http://www.esri.com/software/arcgis/.

[3] Ashdown, Mark, Kenji Oka, and Yoichi Sato. "Combining Head Tracking and Mouse Input for a GUI on Multiple Monitors." *In CHI '05 Extended Abstracts on Human Factors in Computing Systems.* New York, NY, USA: ACM Press, 2005, 1188-1191, DOI=http://doi.acm.org/10.1145/1056808.1056873.

[4] Atlas Gloves. Retrieved on November 16, 2009, from http://atlasgloves.org/.

[5] Bolelli, Levent. "Multimodal Response Generation in GIS." *In Proceedings of the 6th International Conference on Multimodal Interfaces.* New York, NY, USA: ACM Press, 2004, 355-355, DOI=http://doi.acm.org/10.1145/1027933.1028012.

[6] Cabral, Marcio C., Carlos H. Morimoto, and Marcelo K. Zuffo. "On the Usability of Gesture Interfaces in Virtual Reality Environments." *In Proceedings of the 2005 Latin American Conference on Human-Computer Interaction.* New York, NY, USA: ACM Press, 2005, 100-108, DOI= http://doi.acm.org/10.1145/1111360.1111370.

[7] Darrell, Trevor, Konrad Tollmar, Frank Bentley, Neal Checka, Louis-Phillipe Morency, Ali Rahimi, and Alice Oh. "Face-Responsive Interfaces: From Direct Manipulation to Perceptive Presence." *In Proceedings of the 4th international conference on Ubiquitous Computing*. New York, NY, USA: ACM Press, 2002, 135-151.

[8] Andrés del Valle, Ana C. and Jean-Luc Dugelay. "Online Face Analysis: Coupling Head Pose-Tracking with Face Expression Analysis." *In Proceedings of the Tenth ACM International Conference on Multimedia.* New York, NY, USA: ACM Press, 2002, 414-415, DOI=http://doi.acm.org/10.1145/641007.641093.

[9] Entropia Universe. Retrieved on November 17, 2009, from http://www.entropiauniverse.com/.

[10] ezANOVA Statistics Application. Retrieved on October 14, 2009, from http://www.sph.sc.edu/comd/rorden/ezanova/index.html.

[11] Fisher, R. B. and Oliver, P. "Multi-variate Cross-Correlation and Image Matching." *In Proceedings of British Machine Vision Conference*, 1995, 623–632.

[12] Francios, Alexandre R.J., "Real-Time Multi-Resolution Blob Tracking." *IRIS Technical Report IRIS-04-422*, University of Southern California, Los Angeles, April 2004.

[13] Freeman, William. T. and Craig D. Weissman. "Television Control by Hand Gestures." *In Proceedings of the International Workshop on Automatic Face- and Gesture-Recognition.* Zurich, Switzerland: IEEE, 1995, 179-183.

[14] GeoMedia GIS Application from Intergraph. Retrieved on November 12, 2009, from http://www.intergraph.com/.

[15] Google Earth 3D Mapping Application. Retrieved on November 11, 2009, from http://earth.google.com.

[16] Google Earth Developer Blog. Retrieved on November 19, 2009, from http://google-latlong.blogspot.com/2008/02/truly-global.html.

[17] Google Maps 2D Mapping Application. Retrieved on November 19, 2009, from http://maps.google.com.

[18] Gravetter, Frederick. J. and Larry B. Wallnau, *Statistics for the Behavioral Sciences. 6th Edition*. Thomson-Wadsworth Publishing, Belmont, California, United States, 2004, p.124, 244, 432, 650. Print. ISBN 0-534-62203-8.

[19] Gunes, Hatice, Massimo Piccardi, and Tony Jan. "Face and Body Gesture Recognition for a Vision-Based Multimodal Analyzer." *In Proceedings of the Pan-Sydney area workshop on Visual information processing*. ACM, 2004, 19-28.

[20] Hansen, Thomas R., Eva Eriksson, and Andreas Lykke-Olesen. "Use your head: exploring face tracking for mobile interaction." *CHI '06: CHI '06 extended abstracts on Human factors in computing systems*. New York, NY, USA: ACM Press, 2006, 845-850.

[21] Hinckley, Ken, Y. Pausch, John C. Goble, and Neal F. Kassell. “A Survey of Design Issues in Spatial Input.” *In Proceedings of the 7th annual ACM symposium on User interface software and technology.* New York, NY, USA: ACM Press, 1994, 213-222.

[22] Hitwise Weblog. “Google Maps Surpasses Mapquest”. Retrieved on November 19, 2009 from http://weblogs.hitwise.com/heather-dougherty/2009/04/google_maps_surpasses_mapquest.html.

[23] Huang, X. and Y. Lin. “A vision-based hybrid method for facial expression recognition.” *In Proceedings of the 1st international conference on Ambient media and systems.* New York, NY, USA: ACM Press, 2008, Article 4.

[24] Kaneva Online Virtual World. Retrieved on November 17, 2009 from http://www.kaneva.com

[25] Kjeldsen, Rick. “Head Gestures for Computer Control.” *In Proceedings of the IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems.* New York, NY, USA: ACM Press, 2001, 61-68.

[26] Kjeldsen, Rick and Jacob Hartman. “Design issues for vision-based computer interaction systems.” *In Proceedings of the 2001 workshop on Perceptive user interfaces.* New York, NY, USA: ACM Press, 2001, 1-8, DOI= http://doi.acm.org/10.1145/971478.971511.

[27] Kjeldsen, Rick, Anthony Levas, and Claudio Pinhanez. "Dynamically reconfigurable vision-based user interfaces." *In Machine Vision and Applications, Vol.16*, December 2004, 6-12, DOI=10.1007/s00138-004-0145-6.

[28] Kohler, Markus. “Special Topics of Gesture Recognition Applied in Intelligent Home Environments.” *In Proceedings of the International Gesture Workshop on Gesture and Sign Language in Human-Computer Interaction.* London, UK: Springer-Verlag, 1997, 285-296.

[29] Krum, David M., Olugbenga Omoteso, William Ribarsky, Thad Starner, and Larry F. Hodges. “Evaluation of a Multimodal Interface for 3D Terrain Visualization.” *In Proceedings of the conference on Visualization '02.* New York, NY, USA: ACM Press, 2002, 411-418.

[30] Lin, Yuan-Pin, Yi-Ping Chao, Chung-Chih Lin, and Jyh-Horng Chen. "Webcam Mouse Using Face and Eye Tracking in Various Illumination Environments." *In Proceedings of the 27th Annual International Conference of the Engineering in Medicine and Biology Society.* IEEE-EMBS. 2005, 3738-3741.

[31] Loewenich, Frank and Frederic Maire. "Hands-free mouse-pointer manipulation using motion-tracking and speech recognition." *In Proceedings of the 19th Australasian conference on Computer-Human Interaction: Entertaining User Interfaces, Vol. 251.* 2007, 295-302. DOI= http://doi.acm.org/10.1145/1324892.1324955.

[32] MapInfo from Pitney Bowes. Retrieved on November 13, 2009 from http://www.pbinsight.com/products/location-intelligence/applications/mapping-analytical/mapinfo-professional/.

[33] Merdes, Matthias, Jochen Häu, and Matthias Jöst. "'SlidingMap': introducing and evaluating a new modality for map interaction." *ICMI '04: Proceedings of the 6th international conference on Multimodal interfaces*. New York, NY, USA: ACM, 2004, 325-326. DOI=http://doi.acm.org/10.1145/1027933.1027989.

[34] Microsoft Virtual Earth 2D Mapping Application, now known as Bing Maps. Retrieved on November 17, 2009 from http://www.bing/com/maps.

[35] Morency, Louis-Philippe and Trevor Darrell "Head gesture recognition in intelligent interfaces: the role of context in improving recognition." *In Proceedings of the 11th International Conference on Intelligent User Interfaces.* New York, NY, USA: ACM Press, 2006, 32-38, DOI= http://doi.acm.org/10.1145/1111449.1111464.

[36] Morimoto, Carlos H., Yaser Yacoob, and Larry Davis. "Recognition of Head Gestures Using Hidden Markov Models." *In Proceedings of ICPR*. IEEE, 1996, 461-465.

[37] NASA World Wind 3D Virtual Globe Application. Retrieved on November 18, 2009 from http://worldwind.arc.nasa.gov/.

[38] Nummiaro, Katja, Esther K. Meier, and Luc J. Van Gool. "Object Tracking with an Adaptive Color-Based Particle Filter." *Proceedings of the 24th DAGM Symposium on Pattern Recognition*. London, UK: Springer-Verlag, 2002, 353-360.

[39] Oviatt, Sharon, Rachel Coulston, and Rebecca Lunsford. "When do we interact multimodally?: cognitive load and multimodal communication patterns." *In Proceedings of the 6th International Conference on Multimodal Interfaces*. New York, NY, USA: ACM Press, 2004, 129-136.

[40] Panerai, F., S. Hanneton, J. Droulez, and V. Cornilleau-Pérès. "A 6-dof device to measure head movements in active vision experiments: Geometric modeling and metric accuracy." *In Journal of Neuroscience Methods* 90 (1999): 97-106.

[41] Rauschert, Ingmar, Pyush Agrawal, Rajeev Sharma, Sven Fuhrmann, Isaac Brewer, and Alan Maceachren. "Designing a human-centered, multimodal GIS interface to support emergency management." *GIS '02: Proceedings of the 10th ACM international symposium on Advances in geographic information systems*. New York, NY, USA: ACM Press, 2002, 119-124.

[42] Rauschert, Ingmar. "Adaptive multimodal recognition of voluntary and involuntary gestures of people with motor disabilities." *In Proceedings of the 6th International Conference on Multimodal Interfaces.* New York, NY, USA: ACM Press, 2004, 356-356, DOI=http://doi.acm.org/10.1145/1027933.1028013.

[43] Rigoll, Gerhard, Andreas Kosmala, and Stefan Eickeler. "High Performance Real-Time Gesture Recognition Using Hidden Markov Models." *In Proceedings of the International Gesture Workshop on Gesture and Sign Language in Human-Computer Interaction.* New York, NY, USA: ACM Press, 1998, 69-80.

[44] Roweis, S. and Z. Ghahramani. "A unifying review of linear gaussian models." *Neural Computation* 11 (February 1999): 305-345.

[45] Sato, Yoichi, Makiko Saito, and Hideki Koik. "Real-Time Input of 3D Pose and Gestures of a User's Hand and Its Applications for HCI." *In Proceedings of the Virtual Reality 2001 Conference.* New York, NY, USA: ACM Press, 2001, 79-79.

[46] Schöning, Johannes, Brent Hecht, Martin Raubal, Antonio Krüger, Meredith Marsh, and Michael Rohs. "Improving interaction with virtual globes through thinking: helping users ask "why?"." *IUI '08: Proceedings of the 13th International Conference on Intelligent User Interfaces*. New York, NY, USA: ACM Press, 2008, 129-138.

[47] Second Life Virtual World. Retrieved on November 19, 2009 from http://secondlife.com.

[48] Sparacino, Flavia, Christopher Wren, Ali Azarbayejani, and Alex Pentland. "Browsing 3-D spaces with 3-D vision: body-driven navigation through the Internet city." *In Proceedings of the First International Symposium on 3D Data Processing, Visualization, and Transmission.* IEEE, 2002, 224-231.

[49] Sturman, David J. and David Zeltzer. "A Survey of Glove-based Input." *IEEE Computer Graphics and Applications* 14 (1994): 30-39. DOI=10.1109/38.250916.

[50] Valenti, Roberto, Alejandro Jaimes, and Nicu Sebe. "Facial Expression Recognition as a Creative Interface." *In Proceedings of the 13th International Conference on Intelligent User Interfaces.* New York, NY, USA: ACM Press, 2008, 433-434.

[51] Wei, Xiaozhou, Lijun Yin, Zhiwei Zhu, and Qiang Ji. "Avatar-mediated face tracking and lip reading for human computer interaction." *In Proceedings of the 12th Annual ACM International Conference on Multimedia.* New York, NY, USA: ACM Press, 2004, 500-503, DOI=http://doi.acm.org/10.1145/1027527.1027648.

[52] Wii Sports by Nintendo. Retrieved on November 18, 2009 from http://us.wii.com/wiisports/.

[53] XMind Mind Map Software by XMind Ltd. Retrieved on November 19, 2009 from http://www.xmind.net.

[54] Yahoo Maps by Mapquest. Retrieved on November 18, 2009 from http://maps.yahoo.com.

[55] Zhang, Xiaoqin, Weiming Hu, Zixiang Zhao, Yan-guo Wnat, Xi Li, and Qingdi Wei. "SVD based Kalman particle filter for robust visual tracking." *In Proceedings of the 19th International Conference on Pattern Recognition.* IEEE, 2008, 1-4.